Facultad de Ciencias

DISEÑO Y DESARROLLO DE UN ENTORNO DE SIMULACIÓN PARA CLIMATOLOGÍA MARÍTIMA: DE LOS DATOS A LA PREDICCIÓN USANDO HERRAMIENTAS DE CIENCIA DE DATOS

(Design and development of a wave hindcast simulated environment: from data to prediction using data science)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN INGENIERÍA INFORMÁTICA

Autor: Israel Rubio LLarena

Directores: Laureano González Vega
Fernando J. Méndez Incera

Julio - 2019

Agradecimientos

Quiero aprovechar este espacio para agradecer a todas las personas que han estado presentes a lo largo de esta etapa:

En primer lugar, quiero agradecer a mis padres por su apoyo durante todo el máster y su incondicional confianza en mi, incluso cuando yo dudaba de que pudiese conseguirlo.

En segundo lugar, a mi tutor Laureano González, Lalo, por su infinita paciencia y completa dedicación siempre que le he necesitado. También a Fernando Méndez, Fer, por darme la posibilidad de plasmar en código su proyecto, ayudándome si tenía algún problema y motivándome a lo largo de todos estos meses.

Finalmente, acordarme de todos los compañeros, bien del máster (David, Aida, gracias por el apoyo) o del departamento (espero no olvidarme de nadie: Ana, Alba, Albita, Laura, Pablo, Sara, Sonia y, sobre todo, Nico por estar ahí cuando me bloqueaba y solucionar todo con una taza de café).

A todos ellos...¡Muchas gracias!

Resumen

Este documento contiene el trabajo fin de máster (TFM) del Máster Universitario en Ingeniería Informática de la Universidad de Cantabria. Aunque en su mayor parte la información es de carácter informático y computacional, también hace referencia a diversos términos de ingeniería oceanográfica, necesarios para comprender en qué ha consistido el trabajo realizado.

En términos generales, el trabajo realizado consistió en crear un entorno web basado en el lenguaje de programación Python que permita la simulación de series de cientos de miles de estados de oleaje que se propagan con un sistema híbrido compuesto por un modelo numérico de oleaje, un algoritmo matemático de selección y, posteriormente, otro algoritmo de interpolación para conocer las condiciones resultantes en el punto deseado. También se realizó el diseño y puesta en marcha del equipo hardware necesario, la automatización de las tareas de seguridad y backup y el despliegue en internet:

http://irl167.pythonanywhere.com/

El tema puede parecer sencillo pero su complejidad es alta ya que, para su resolución con cierta precisión, se requiere de series históricas de variables geofísicas suficientemente extensas para poder ser caracterizadas de forma estadística, llegando a tener archivos con 35 años de datos evaluados cada hora y de los cuales se precisa, para conocer el estado de mar, un mínimo de tres variables: altura de ola significante (Hs), periodo medio (Tm) y dirección media del oleaje (θ).

Las técnicas matemáticas y computacionales empleadas para la predicción de resultados han sido el algoritmo de máxima disimilitud (MDA) para la selección de casos óptimos y la validación cruzada de $k-$iteraciones combinada con las funciones de base radial (RBF) como método de interpolación posterior. Luego, para su despliegue, se ha empleado Flask como framework y distintos módulos de Javascript para conseguir todas las funcionalidades requeridas.

Palabras clave: TFM, ingeniería informática, ingeniería oceanográfica, Universidad de Cantabria, Python, MDA, RBF, validación cruzada de $k-$iteraciones.

Abstract

This document contains the final project of Master's Degree in computing engineering at University of Cantabria. It uses mainly the computational and mathematical language, but it also includes some references to oceanographic engineering to improve the knowledge of the whole process.

In general terms, the main job was the development of a web-site in Python which simulates the hybrid propagation of the wave hindcast using mathematical algorithms to formerly select data from a statistical database and finally interpolate the results in the desired point. Besides that, it was designed a hardware structe and planned an automatization program to keep it safe. The website was deployed in the domain:

http://irl167.pythonanywhere.com/

Although it can seem vain, it is a high complexity problem cause it needs long temporal series to obtain accurate results; the largest experiments has involved 35 years of statistical data where we obtained hourly three variables: significant wave length (Hs), mean period (Tm) and mean direction (θ).

The mathematical and computational techniques applied in prediction were maximum dissimilarity algorithm (MDA) for selecting cases and $k-$fold cross-validation in addition to radial basis functions (RBF) to interpolate.After that, it was deployed using Flask framework and different JavaScript extensions to fulfil the desired objectives.

Keywords: Wave hindcast, Computing Engineering, Oceanographic Engineering, University of Cantabria, Python, MDA, RBF, $k-$fold cross validation.

Índice general

Resumen **I**

Abstract **III**

Lista de figuras **VII**

Lista de tablas **IX**

1. Introducción **1**
1.1. Motivación . . . 1
1.2. Aptitudes que se han desarrollado . . . 2
Ingeniería de software . . . 2
Matemáticas y Computación . . . 2
Redes y Sistemas . . . 2
Diseño y Evaluación . . . 2
Tecnologías para el desarrollo de aplicaciones empresariales en internet 2
Análisis de datos . . . 2
Certificación de la calidad y seguridad de sistemas informáticos . . . 2

2. Técnicas **3**
2.1. Lenguajes de programación . . . 3
CSS . . . 3
HTML . . . 3
JavaScript . . . 3
Python . . . 4
2.2. Tecnología y herramientas . . . 5
Apache . . . 5
Base de datos de CSIRO . . . 5
Flask . . . 5
LaTex . . . 6
Leaflet . . . 6
Navegador web . . . 6
Pace . . . 6
PyCharm . . . 6
SonarQube . . . 7
El modelo SWAN . . . 7
2.3. Algoritmos matemáticos . . . 7
Validación cruzada . . . 7
Algoritmo de máxima disimilitud (MDA . . . 9
Interpolación por funciones de base radial (RBF) . . . 11

3. Desarrollo **13**
3.1. Análisis y especificación de requisitos . . . 13
a) Requisitos funcionales . . . 14
b) Requisitos no funcionales . . . 15
c) Casos de uso . . . 16
3.2. Diseño . . . 18
a) Diseño arquitectónico . . . 18
b) Diseño detallado . . . 18
c) Diseño desarrollado . . . 19
3.3. Implementación . . . 26
NAS . . . 26
RAID . . . 29
Despliegue . . . 31
3.4. Pruebas . . . 31
3.5. Mejoras . . . 32
3.6. Análisis de calidad . . . 33
Primera iteración . . . 33
Segunda iteración . . . 34
Proyecto final . . . 35
3.7. Seguridad y Backup . . . 36

4. Conclusiones **37**
4.1. Demostración . . . 37
4.2. Resultados obtenidos y su explotación . . . 48
4.3. Contribución actual . . . 50
4.4. Posibilidades de futuro . . . 50

Anexos **a**

A. Scriptboard inicial **a**

B. Correcciones después de pruebas **c**

C. Distribución del servidor **g**

D. Contenido del DVD **i**

Bibliografía **k**

Índice de figuras

1. Previsión de zonas de inundación de Majuro en 2055. 1

2. Nodos de reanálisis de oleaje de CSIRO el las islas del Pacífico. 5
3. Validación cruzada dejando uno fuera. 8
4. Validación cruzada $k-$fold con 5 iteraciones. 9
5. Función de aproximación a partir de varias funciones obtenidas por RBF. 11

6. Esquema del proceso de desarrollo de software. 13
7. Scriptboard vs Diseño web final. 14
8. Esquema de requisitos no funcionales de Sommerville. 15
9. Esquema de requisitos no funcionales según la norma ISO 25.000. . . . 15
10. Diagrama de despliegue. 18
11. Esquema de desarrollo TDD. 19
12. Código de MDA en Python (eliminando primer valor). 21
13. Comparación del código de Matlab frente a librería en Python. 22
14. Primera versión de la página de inicio (mapa global). 23
15. Primera versión de la página de selección de puntos. 24
16. Primera versión de la página de MDA. 25
17. Primera versión de la página de SWAN. 25
18. Primera versión de la página de RBF. 25
19. Primera versión de la página de información del proyecto con cabecera. 26
20. Diskstation DS418J. 27
21. Escritorio de File station. 27
22. Synology Quick Connect. 28
23. Página de acceso al NAS. 28
24. RAID 0. 29
25. RAID 1. 29
26. RAID 5. 29
27. RAID 6. 30
28. RAID 10. 30
29. Grupos de almacenamiento del NAS. 30
30. Colormap. 33
31. Resultados de Sonarqube del primer prototipo. 33
32. Resultados de Sonarqube del primer prototipo sin virtualenv. 34
33. Resultados de Sonarqube del segundo prototipo. 34
34. Bug del segundo prototipo. 35
35. Resultados de Sonarqube del proyecto final. 35
36. Módulo de seguridad del NAS. 36
37. Informe de seguridad mensual. 36

38. Ejecución de Flask. 37
39. Página de inicio. 37
40. Carga de datos a nivel del mapa regional. 38
41. Mapa regional. 38
42. Carga de datos a nivel del mapa local. 39
43. Mapa regional. 39
44. Ejecución del MDA. 39
45. MDA completado. 40
46. Datos obtenidos del MDA. 40
47. Gráficas temporales del MDA. 41
48. Gráficas de dispersión del MDA. 41
49. Inicialización de SWAN. 42
50. Puntos cercanos a la costa. 42
51. Casos de SWAN. 42
52. Tabla de casos SWAN. 43
53. Iteraciones de SWAN. 43
54. Casos resueltos en SWAN. 44
55. Elección del caso SWAN. 44
56. Caso seleccionado de SWAN. 44
57. Mapa de color para el caso 2. 45
58. Mapa de color ampliado. 45
59. Inicio de RBF. 46
60. RBF completado. 46
61. Datos obtenidos de RBF. 47
62. Gráfica del punto p1. 47
63. Gráfica de tiempos de procesado por caso. 49
64. Jupyter notebook de Roi namur. 50

Índice de cuadros

1. Tabla de requisitos funcionales. 14
2. Tabla de requisitos no funcionales. 16
3. Tabla de casos de uso. 17

4. Tabla de resultados. 48

Siglas y Acrónimos

A continuación se incluye un diccionario de términos, siglas y acrónimos usados en este documento:

COBIT

Objetivos de control de la información, acrónimo en inglés de Control Objectives for Information and related Technology.

CSIRO

Organización de investigación científica e industrial de la Commonwealth, acrónimo en inglés de Commonwealth Scientific and Industrial Research Organisation.

CSS

Lenguaje de diseño gráfico hojas de estilo en cascada, acrónimo en inglés de Cascading Style Sheet.

HDF5

Formato de datos jerárquico, acrónimo en inglés de Hierarchical Data Format version 5.

HTML

Lenguaje de marcas de hipertexto, acrónimo en inglés de HyperText Markup Language.

HTTP

Protocolo de transferencia de hipertexto, acrónimo en inglés de HyperText Transfer Protocol.

IDE

Entorno de desarrollo integrado, acrónimo de su forma en inglés Integrated Development Environment.

JS

Lenguaje de programación Javascript.

KISS

Metodología de desarrollo basada en la simplicidad, acrónimo de su forma en inglés Keep It Small and Simple.

MDA

Algoritmos de máxima disimilitud, acrónimo de su forma en inglés Maximum Dissimilarity Algorithm.

MVC

Patrón de diseño arquitectónico Modelo-Vista-Controlador.

NaN

Expresión para indicar que el valor no es un número, acrónimo de su forma en inglés Not a Number.

NAS

Sistema de almacenamiento conectado a la red, acrónimo de su forma en inglés Network Attached Storage.

RAID

Matriz redundante de discos independientes, acrónimo de su forma en inglés Redundant Array of Independent Disks.

RBF

Método de función de base radial o función radial básica (ambos términos son admitidos), acrónimo de su forma en inglés Radial Basis Function.

SWAN

Modelo numérico de propagación de oleaje, acrónimo de su forma en Simulating WAaves Nearshore.

TDD

Modelo de diseño ágil basado en el desarrollo guiado por pruebas, acrónimo de su forma en inglés Test Driven Development.

TFM

Trabajo fin de máster.

WSGI

Interfaz de enlace de servidores web, acrónimo de su forma en inglés Web Server Gateway Interface.

Capítulo 1

Introducción

1.1. Motivación

Este TFM se realizó en colaboración entre el Departamento de Matemáticas, Estadística y Computación, bajo la tutela del profesor Laureano González Vega, y la E.T.S. de Ingenieros de Caminos, Canales y Puertos - Departamento de Ciencias y Técnicas del Agua y del Medio Ambiente, codirigido por el profesor Fernando Méndez Incera, en el desarrollo de un entorno de simulación para el estudio del oleaje en islas del Pacífico. En el desarrollo del mismo, se toma como base las tesis doctorales de Paula Camus [5] y Cristina Izaguirre [15].

La zona de estudio ha sido elegida por ser significativo el riesgo de inundación de Majuro [30], capital de las Islas Marshall, tal y como se muestra en la previsión para el año 2055 (ver figura a continuación); de ahí la necesidad de un modelo para zonas cercanas a la costa y que no requiera software de terceras partes.

Figura 1: Previsión de zonas de inundación de Majuro en 2055.

Por tanto, el objetivo de este proyecto es crear un entorno de simulación para climatología marítima, basado en modelos previos como MUSCLE [25], que genere de forma recurrente modelos de reanálisis del oleaje en aguas someras (aquellas que tienen una profundidad reducida) y que permita la propagación mediante modelos numéricos de los estados de mar desde un punto frente a la costa hasta otro de estudio por medio de herramientas de Ciencia de Datos [3].

1.2. Aptitudes que se han desarrollado

Ingeniería de software

Se ha desarrollado un ciclo de trabajo para el despliegue del entorno de computación más adecuado a las necesidades del proyecto, eligiendo desde el framework más acorde (Flask) hasta el lenguaje de programación (Python).

Matemáticas y Computación

Se han adaptado e implementado los algoritmos necesarios para la predicción de las condiciones meteorológicas costeras, empleando un algoritmo de selección de casos (MDA) y otro de interpolación (validación cruzada $k-$fold y RBF).

Redes y Sistemas

Se ha montado un servidor autónomo que incluye las bases de datos, el entorno web y permite realizar copias de seguridad automáticas de la información.

Diseño y Evaluación

Se ha probado el sistema y realizado las mejoras oportunas para facilitar su uso.

Tecnologías para el desarrollo de aplicaciones empresariales en internet

Se han integrado los modelos que se encontraban actualmente en formato Matlab al entorno web programándolos de nuevo en Python y se ha adaptado código existente en Python a las necesidades actuales.

Análisis de datos

Se ha comprobado el grado con que los resultados teóricos obtenidos concuerdan con los datos reales y el número de casos óptimo o más adecuado según el tiempo de ejecución.

Certificación de la calidad y seguridad de sistemas informáticos

Se ha comprobado que el sistema cumple con los estándares de calidad, empleando herramientas de análisis estático de código fuente, y de seguridad.

Capítulo 2

Técnicas

2.1. Lenguajes de programación

Los lenguajes de programación usados en este documento son:

CSS

CSS es un lenguaje de hojas de estilo creado para definir y controlar la presentación de un documento electrónico creado en lenguaje HTML o XHTML. Permite ajustar las características de las capas, colores y fuentes en una hoja de estilo, común para todos los documentos HTML, reduciendo así la complejidad del código y evitando repeticiones. Fue desarrollado por CSS Working Group y se encuentra actualmente en su versión 2.2 (versión del 12 de abril de 2016).

HTML

HTML es un lenguaje de programación utilizado en el desarrollo de páginas web de internet. Fue creado en 1990 por Tim Berners-Lee con el objetivo de facilitar el acceso a documentos de investigación a científicos de distintas universidades. Es el estándar aceptado por el World Wide Web Consortium (W3C), organización que se encarga de estandarizar todas las tecnologías web. Actualmente, la versión más extendida es HTML5 (versión del 28 de octubre de 2014).

JavaScript

JavaScript o JS es un lenguaje ligero (que ocupa poco espacio y fácil de editar), interpretado (la traducción a lenguaje máquina se hace instrucción a instrucción y no se guarda el resultado), orientado a objetos (los objetos manipulan los datos de entrada para obtener datos específicos de salida) y con funciones de primera clase (las funciones en ese lenguaje son tratadas como cualquier otra variable). Se usa principalmente para páginas web, pero también es usado en muchos entornos sin navegador, tales como node.js, Apache CouchDB y Adobe Acrobat. Ha sido desarrollado por Oracle y el estándar actual es el ECMAScript 2019 (versión del 29 de enero de 2019).

Python

Python es un lenguaje de programación orientado a objetos (los objetos manipulan los datos de entrada para obtener datos específicos de salida), interpretado (convierte el código escrito a lenguaje máquina a medida que es ejecutado, no precisando compilación previa) y de alto nivel (expresa los algoritmos de forma similar al lenguaje humano para mejorar su comprensión). Es gratuito y permite ser usado con los principales sistemas operativos del mercado. La versión más reciente, empleada en la realización de este trabajo, es la v3.7.3 (versión del 25 de marzo de 2019). Entre las librerías que se han empleado en este proyecto, destacan:

P1.Pandas

Pandas es una extensión de Python de uso libre para el análisis de alto nivel y el posterior estudio de datos de forma sencilla en Python. Forma parte del proyecto NumFOCUS y se encuentra actualmente en su versión v0.23.4 (versión del 3 de agosto de 2018).

P2.NumPy

NumPy es una extensión de Python para el uso de vectores y matrices, pero lo que la convierte en la biblioteca de funciones matemáticas de alto nivel más usada es que resulta de gran utilidad al trabajar con ficheros procedentes de Matlab ya que permite usar muchas de las funcionalidades de dicho programa en Python. Forma parte también del proyecto NumFOCUS y se encuentra actualmente en su versión v1.17 (versión del 30 de mayo de 2019).

P3.Scipy

SciPy es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos, científicos e ingenieriles. Forma parte también del proyecto NumFOCUS y se encuentra actualmente en su versión v1.16.4 (versión del 28 de mayo de 2019).

P4.Matplotlib

Matplotlib es una extensión de Python para la generación de gráficos 2D a partir de datos contenidos en listas o arrays. Se encuentra actualmente en su versión v3.1 (versión del 18 de mayo de 2019).

P5.NetCDf4

NetCDf4 es un módulo libre para Python usado para la lectura de archivos netCDF, generalmente identificados con la extensión .nc, que permiten la creación de archivos de datos basados en arrays en formato de datos HDF5. Son el formato más usado en climatología, meteorología y oceanografía. Forma parte de un proyecto de la University Corporation for Atmospheric Research (UCAR) y se encuentra actualmente en su versión v1.5.1.2 (versión del 6 de mayo de 2019).

2.2. Tecnología y herramientas

Las herramientas utilizadas en el proceso han sido:

Apache

El servidor HTTP Apache es un servidor de código abierto que soporta distintas plataformas (GNU/Linux, Windows y MacOS) y que implementa el protocolo HTTP/1.1 según la normativa RFC2616. Es el servidor más usado por su versatilidad: es fácilmente configurable, es modular y permite distintos formatos de bases de datos. Fue desarrollado por Apache Software Foundation y se encuentra actualmente en su versión v2.4.37 (versión del 23 de octubre de 2018).

Base de datos de CSIRO

La base de datos de CSIRO incluye datos meteorológicos desde enero de 1979 hasta diciembre de 2018. Para este trabajo se han utilizado los datos referentes a las batimetrías más cercanas a la isla de Majuro, capital de las Islas Marshall [31].

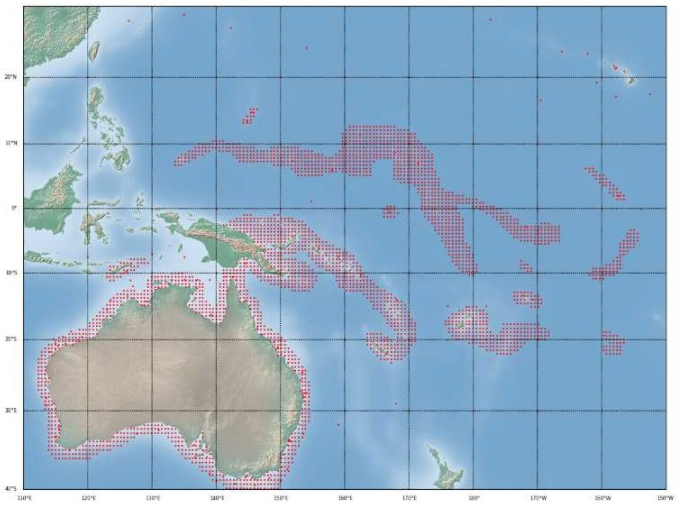


Figura 2: Nodos de reanálisis de oleaje de CSIRO el las islas del Pacífico.

Dado el gran volumen de datos de cada nodo (para cada punto hay datos de altura de ola, frecuencia y dirección media y de cada una de sus componentes, desde 1979 hasta 2018 medidos cada hora) se decidió descargar varios nodos de estudio y comprimirlos en formato NetCDf4 para agilizar el proceso de cálculo.

Flask

Flask es un framework de Python que permite crear aplicaciones web de forma fácil y rápida con pocas líneas de código. Está basado en la especificación WSGI de Werkzeug (interface simple y universal entre servidores y frameworks o aplicaciones web, es el estándar de Python). Se encuentra actualmente en su versión v1.0.2 (versión del 8 de febrero de 2018).

LaTex

LaTex es un sistema de composición de textos, orientado a la creación de documentos escritos de alta calidad tipográfica. Es usado principalmente en la generación de artículos y trabajos científicos [20]. LaTeX fue programado por Leslie Lamport en 1984 para facilitar el uso del lenguaje de composición tipográfica. Para la realización de este trabajo se empleó LaTex utilizando TeXworks v0.6.2 como front-end.

Leaflet

Leaflet es una librería de código abierto para JavaScript para la creación de mapas interactivos para internet. Permite importar datos desde archivos NetCDf4. Se encuentra actualmente en su versión v1.0.2 (versión del 26 de octubre de 2017).

Navegador web

Un navegador web es una aplicación que permite el acceso a contenidos de internet localizados en un sitio web, comúnmente denominado página web. El primer navegador fue desarrollado en 1990 por Tim Berners-Lee del CERN, pero solo funcionaba en estaciones NeXT. Desde entonces su uso se ha popularizado y existen distintos programas según el entorno. Para la realización de este trabajo se hicieron pruebas en los siguientes navegadores:

- Microsoft Edge: navegador por defecto de Windows, es una versión mejorada de Microsoft Internet Explorer, actualmente se encuentra en su versión v44.17763.1 (2 de octubre de 2018).
- Apple Safari: navegador por defecto de MacOS, actualmente se encuentra en su versión v12.1.1 (13 de mayo de 2019).
- Mozilla Firefox: navegador por defecto de Linux Ubuntu, actualmente se encuentra en su versión v67.0.3 (18 de junio de 2019).
- Google Chrome: navegador web desarrollado por Google, gratuito y de código abierto, con motor de renderizado de WebKit. Es el navegador más usado hoy en día por lo que se decidió por incluir en las pruebas pese a no ser el navegador por defecto en ningún entorno de escritorio (aunque sí para móviles Android). Actualmente se encuentra en su versión v75.0.377 (18 de junio de 2019).

Pace

Pace es una biblioteca de Javascript y CSS que permite de forma automática añadir barras de progreso o indicadores de actividad cuando se carga una página web o se inicia un proceso en Ajax. Es software libre y open source. Se encuentra actualmente en su versión v1.5.1 (versión del 8 de mayo de 2019).

PyCharm

PyCharm es un IDE de Python para desarrollo profesional. Fue el entorno elegido para el desarrollo porque permitía el uso de Django y Flask, los dos frameworks más comunes en Python. Se encuentra actualmente en su versión v2019.1 (versión del 1 de marzo de 2019).

SonarQube

SonarQube es una plataforma libre para el análisis del código fuente. Usa diversas herramientas de análisis estático, tales como Checkstyle (comprueba si el código Java se adapta al estándar), PMD (análisis estático de la integración del código) o FindBugs (herramienta de debug), para obtener métricas que mejoren la calidad del código analizado. Se encuentra actualmente en su versión v7.7 (versión del 20 de marzo de 2019).

El modelo SWAN

El modelo SWAN es un programa libre de simulación de propagación basado en el modelo de onda de energía en coordenadas cartesianas (Booij et al., 1999 [4]). Forma parte de un proyecto de Delft University of Technology y se encuentra actualmente en su versión v41.31 (versión del 28 de mayo de 2019).

El motivo de su uso es que en aguas profundas las olas no son afectadas por la batimetría, pero en aguas someras o poco profundas (que es el caso de estudio) se producen variaciones en la altura y dirección de la ola, por lo que se precisa de un método que tenga en cuenta los procesos de transformación (refracción, y asomeramiento debido a las corrientes, difracción de forma abrupta y pérdida de energía por disipación cerca del fondo).

Este modelo numérico [17] puede ser resuelto con apenas entre 10 y 100 iteraciones por periodo de ola, aunque limitado a pequeñas áreas (de entre 1 y 10 km), definiendo la ola cada hora por su altura de ola significante (Hs), periodo medio (Tm) y dirección media del oleaje (θ). La ecuación empleada para resolver el modelo de propagación de oleaje en SWAN es una ecuación de transporte de energía espectral. Si consideramos 4 dimensiones, dos espaciales (x, y), la frecuencia (σ) y la dirección (θ), la ecuación sería:

$$\frac{\partial N}{\partial t} + \frac{\partial c_x N}{\partial x} + \frac{\partial c_y N}{\partial y} + \frac{\partial c_\sigma N}{\partial \sigma} + \frac{\partial c_\theta N}{\partial \theta} = \frac{S}{\sigma} \tag{2.1}$$

En la parte izquierda de la ecuación, el primer término representa la variación frente al tiempo, los términos segundo y tercero la velocidad de propagación en el espacio geográfico, el cuarto término el cambio en la frecuencia relativa y el quinto término la refracción inducida por la profundidad y las corrientes. En la parte derecha se representan las fuentes y sumideros de la energía espectral, como pueden ser la generación de oleaje por viento, la disipación de energía por fricción con el fondo y rotura del oleaje y las interacciones no lineales del oleaje.

2.3. Algoritmos matemáticos

Los algoritmos usados en este documento son:

Validación cruzada

La validación cruzada o cross-validation [16] es una técnica matemática de evaluación de resultados que permite garantizar que los datos empleados son independientes de la partición utilizada entre entrenamiento (training) y prueba (test). Existen tres tipos fundamentales de validación cruzada:

1. Validación cruzada dejando uno fuera (leave-one-out): supuesto un conjunto de $N-$datos, tomamos un único dato de prueba y empleamos los $N-1$ datos restantes para el entrenamiento. Se repite el proceso con el mismo número de iteraciones que muestras, por lo que es un proceso muy efectivo, pero a la vez muy costoso. Este proceso es el que se emplea originalmente en el algoritmo de Rippa [24].

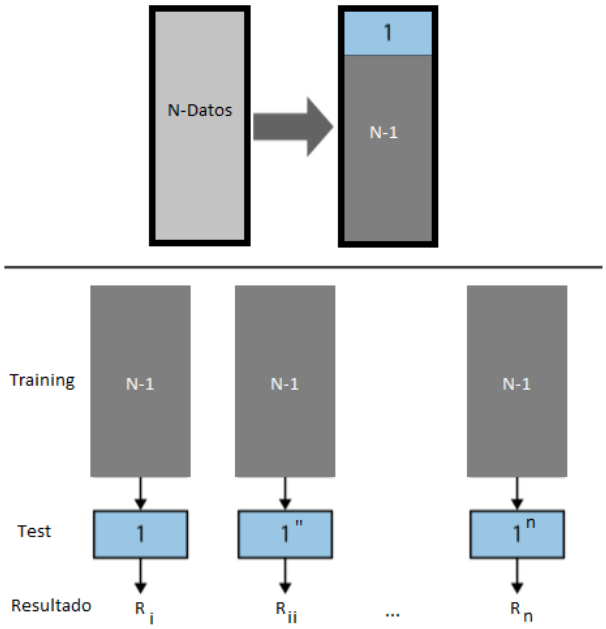


Figura 3: Validación cruzada dejando uno fuera.

2. Validación cruzada aleatoria: supuesto un conjunto de $N-$datos, se divide aleatoriamente el conjunto entre los datos de entrenamiento y prueba, repitiendo el proceso $k-$veces. El resultado final se corresponde a la media aritmética de los valores obtenidos para las diferentes divisiones. La ventaja de este método es que la división de datos entre entrenamiento y prueba no depende del número de iteraciones, pero tiene el gran inconveniente de que, con este método, hay muestras que se quedan sin evaluar y otras se evalúan más de una vez, es decir, los subconjuntos de prueba y entrenamiento se pueden solapar.

3. Validación cruzada de $k-$iteraciones ($k-$fold): supuesto un conjunto de $N-$datos, se toma el conjunto de los datos originales y se crea a partir de ellos dos subconjuntos: uno de entrenamiento y otro de validación. Después se divide el conjunto de entrenamiento en k subconjuntos y se toma cada subconjunto k como conjunto de prueba del modelo mientras que el resto de los datos se tomará como conjunto de entrenamiento. Este proceso se repetirá k veces y en cada iteración se seleccionará un conjunto de prueba diferente. Una vez finalizadas las iteraciones, se calcula la precisión y el error para cada uno de los modelos producidos y, para obtener la precisión y el error final, se calcula el promedio de los k modelos entrenados. éste es el método elegido para nuestra validación en funciones de base radial (RBF) con un subconjunto $k = 5$:

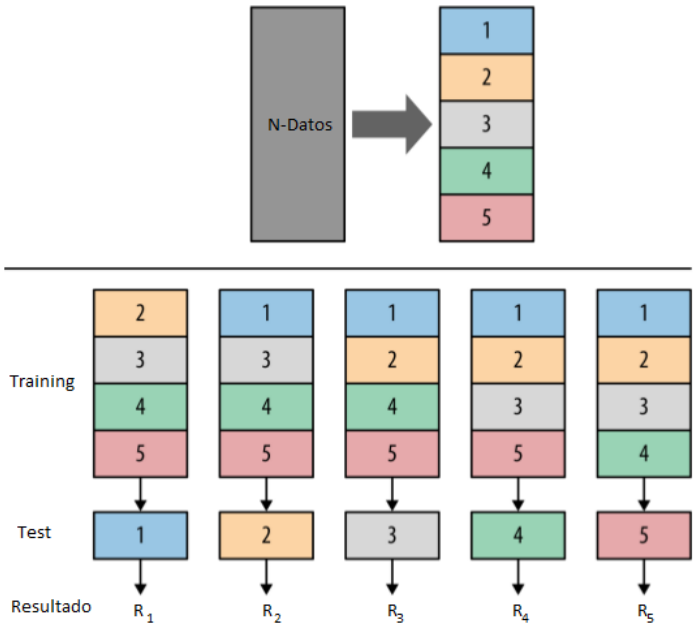


Figura 4: Validación cruzada $k-$fold con 5 iteraciones.

Algoritmo de máxima disimilitud (MDA

El objetivo de cualquier algoritmo de selección es la obtención de un subconjunto de datos representativo de tamaño M a partir de una base de datos de tamaño N, siendo N mayor que M. El algoritmo MDA fue descrito por Kennard y Stone en 1969 y surgió en la búsqueda de métodos computacionales para la selección de compuestos químicos en bases de datos de compuestos muy extensas (Snarey et al., 1997 [26]), aunque puede ser aplicados a otros ámbitos (como sucede en nuestro caso).

Dado un conjunto de datos $X = \{x_1, x_2, ..., x_N\}$, consistente en N vectores $n-$dimensionales, se pretende obtener un subconjunto M de vectores $\{v_1, v_2, ..., v_M\}$ representativo del conjunto original mediante el algoritmo MDA. La selección comienza inicializando el subconjunto al transferir un único vector $\{v_1\}$ desde el conjunto original y usar los $M-1$ elementos restantes para calcular, de forma iterativa, su disimilitud con el vector seleccionado. Cuando se encuentra el vector $\{v_2\}$, que presenta máxima disimilitud, se transfiere al subconjunto M y se comienza de nuevo el proceso iterativo frente a los $M-2$ vectores restantes. El proceso termina al alcanzar las M iteraciones.

Este algoritmo admite distintas variaciones en función del criterio escogido para la inicialización del subconjunto $\{v_1\}$:

- Se puede escoger de forma aleatoria.
- Se puede escoger el elemento de mayor disimilitud de la base de datos de partida.
- Se puede escoger el elemento más próximo al centro de la base de datos.

Una vez seleccionado el primer elemento del subconjunto, la selección del resto de elementos se realiza en dos fases:

1. Para cada dato que aún forma parte de la base de datos de partida se calcula la distancia o disimilitud con el resto de elementos del subconjunto y se anota dicha distancia.

2. Una vez se dispone de todas las distancias entre el dato de partida y el subconjunto, se elige aquel dato cuya distancia sea el valor máximo obtenido.

Por ejemplo, para un dato R, tal que $R \leq M$, primero se calcula la distancia entre el dato i de la muestra $N - R$ y los j elementos del subconjunto R:

$$d_{i,j} = \parallel x_i - v_j \parallel; i = 1, ..., N; j = 1, ..., R \tag{2.2}$$

Posteriormente se calcula la distancia $d_{i,subconjunto}$ entre el dato i y el subconjunto R:

$$d_{i,j} = \left\{ \begin{array}{c} \text{mín} \\ \text{máx} \\ \text{mean} \\ \text{sum} \end{array} \right\} \{\parallel x_i - v_j \parallel\}; i = 1, ..., N; j = 1, ..., R \tag{2.3}$$

Una vez calculadas las $N - R$ distancias, se selecciona el dato de máxima disimilitud para ser incluido en el subconjunto y el proceso continua iterativamente.

Para nuestro caso, al ser las variables empleadas muy diferentes entre sí, es preciso realizar previamente un proceso de preselección, posteriormente normalización, calcular a continuación el MDA y, finalmente, desnormalizar el resultado.

Preselección

El proceso de preselección consiste en dividir el espacio de los datos de entrada en intervalos equiespaciados a lo largo de las 3 variables: altura de ola significante (Hs), periodo medio (Tm) y dirección media del oleaje (θ). Por tanto, los N datos de entrada: $X_{(i)} = \{H_{s(i)}, T_{m(i)}, \theta_{m(i)}\}\}$; $\quad i = 1, ..., N$ se reducen a un conjunto de P vectores: $X_{(i)} = \{H_{s(i)}, T_{m(i)}, \theta_{m(i)}\}\}$; $\quad i = 1, ..., P$.

Normalización

En la normalización debemos tener en cuenta qué variables son escalares (H_s y T_m) y cúales son circulares (θ_m). Las variables escalares se normalizan entre $[0, 1]$ a través de una transformación lineal una vez conocidos sus valores máximo y mínimo en el conjunto de datos, mientras que para las circulares debemos tener en cuenta que la distancia máxima entre 2 puntos de la circunferencia es igual a π. De este modo quedan normalizadas las tres variables:

$$H_{s(i)}^{norm} = \frac{H_{s(i)} - H_s^{min}}{H_s^{max} - H_s^{min}} \tag{2.4}$$

$$T_{m(i)}^{norm} = \frac{T_{m(i)} - T_m^{min}}{T_m^{max} - T_m^{min}} \tag{2.5}$$

$$\theta_{m(i)}^{norm} = \frac{\theta_{m(i)}}{\pi} \tag{2.6}$$

Selección

Éste es el proceso de aplicación de las técnicas de selección de MDA propiamente dicho. Tiene la dificultad añadida de que la distancia entre las variables circulares no puede ser determinada a través de la distancia euclídea debido a la escala de valores continua en la circunferencia. Si consideramos que los centroides obtenidos por el algoritmo de MDA son $D_j = \{H_{s(j)}, T_{m(j)}, \theta_{m(j)}\}; j = 1, ..., M$, siendo M el número de centroides, la distancia implementada será:

$$\parallel X_{(i)} - D_j \parallel = \sqrt{\left(H_{s(i)} - H_{s(j)}\right)^2 + \left(T_{m(i)} - T_{m(j)}\right)^2 + \Delta^2} \tag{2.7}$$

$$\Delta = \left(min\left(\parallel \theta_{m(i)} - \theta_{m(i)} \parallel, 2 - \parallel \theta_{m(i)} - \theta_{m(i)} \parallel\right)\right) \tag{2.8}$$

Desnormalización

El proceso de desnormalización es el opuesto al de normalización. Es el último paso del método y permite devolver las variables a su escala original:

$$H_{s(j)}^{desnorm} = H_{s(j)} \cdot \left(H_s^{max} - H_s^{min}\right) + H_s^{min} \tag{2.9}$$

$$T_{m(j)}^{desnorm} = T_{m(j)} \cdot \left(T_m^{max} - T_m^{min}\right) + T_m^{min} \tag{2.10}$$

$$\theta_{m(j)}^{desnorm} = \theta_{m(j)} \cdot \pi \tag{2.11}$$

Interpolación por funciones de base radial (RBF)

El objetivo de las técnicas de interpolación es la obtención de una función continua a partir de su valor para una serie determinada de puntos, infiriéndose el valor de la función en cualquier otro punto. El problema se complica cuando el sistema es multivariante, es decir, en un espacio multidimensional; situación en la que la función de base radial RBF es un método de interpolación bastante efectivo: para una determinada región n-dimensional del espacio euclídeo, RBF nos permite obtener una función continua a partir de una serie m de puntos $X = \{x_1, ..., x_M\}$, pudiéndose posteriormente inferir el valor de la función multivariante $f(k)$, siendo k cualquier otro punto de la región considerada (Hastie et al. [13]).

Esta técnica considera que la función de aproximación se obtiene como una combinación lineal de una serie de funciones de base radial simétricas básicas localizadas, definidas por la expresión $\Phi(\parallel \cdot \parallel)$, siendo $\parallel \cdot \parallel$ una norma euclídea y, por tanto, sólo dependiente de la distancia euclídea.

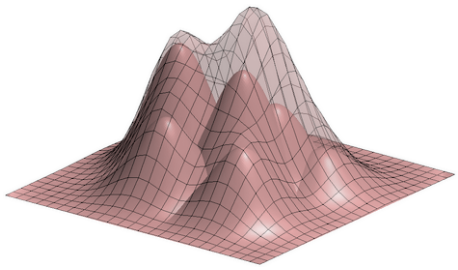


Figura 5: Función de aproximación a partir de varias funciones obtenidas por RBF.

Por tanto, dado el conjunto de puntos $X = \{x_1, ..., x_M\} \in \mathbb{R}^n$, que tienen valores reales asociados a la función $f_i = f(x_i) \in \mathbb{R}$, siendo $i = 1, ..., N$, el objetivo del método RBF es obtener una función de aproximación $f : \mathbb{R}^n \to \mathbb{R}$ de la forma:

$$f(x)^{RBF} = p(x) + \sum_{i=1}^{N} a_i \Phi(\| x - x_i \|), \qquad x \in \mathbb{R}^n \tag{2.12}$$

En donde:

- $p(x)$ es un polinomio que incluye todas las variables de estudio, en principio formada por $N-$monomios de grado uno, siendo N la dimensionalidad de los datos; para nuestro caso: $p(x) = b_0 + b_1 H_i + b_2 T_i + b_3 \theta_i$
- a_i son los coeficientes de ajuste de RBF.
- Φ es la función de base radial.
- x_i son los centros de la aproximación RBF.

Al ajustar la función RBF con centro en los puntos x_i, se interpola el vector de datos a $f = \{f_1, ..., f_N\}$ y queda definida por los coeficientes de ajuste a_i y los coeficientes del polinomio b_i, pudiendo expresarse como una matriz:

$$\begin{pmatrix} A_{i,j} & P_{i,j} \\ P^T & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{2.13}$$

En donde:

- $A_{i,j} = \Phi$ ($\| x$ - $x_i \|$), siendo$i, j = 1, ..., N$.
- $P_{i,j} = p_j(x_i)$, siendo $i = 1, ..., N$ y $j = 1, ..., M$.

Por tanto, pese a la complejidad formal, todo se reduce a resolver un sistema de ecuaciones lineales.

Función radial gaussiana

Aunque existen distintos tipos de funciones radiales (lineal, cúbica, multicuadrática, spline,...), en nuestro caso sólo empleamos la función gaussiana:

$$\Phi(x) = e^{-\frac{r^2}{2c^2}} \tag{2.14}$$

En donde c es un parámetro definido por el usuario y cuyo valor óptimo depende de la cantidad y de la distribución de los puntos disponibles de partida y del vector con el valor de la función f en los puntos considerados, tal y como se define en el algoritmo de Rippa [24].

Capítulo 3

Desarrollo

Es éste un proyecto de ingeniería de software: el desarrollo se realiza de forma iterativa e incremental, de forma que en cada iteración se realice el análisis de requisitos, el diseño funcional, la implementación práctica y una fase de pruebas y planteamiento de mejoras, hasta obtener el resultado óptimo. Este proceso se basa en las premisas de desarrollo de software de Cockburn [8]:

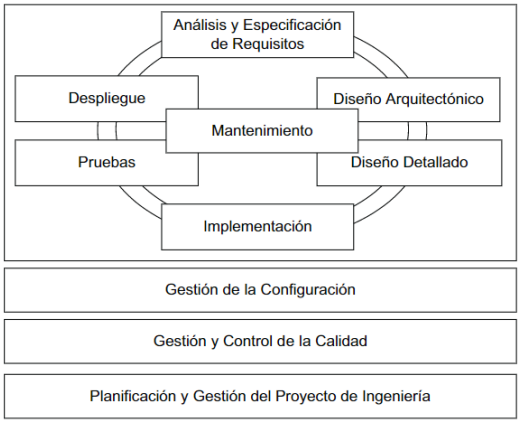


Figura 6: Esquema del proceso de desarrollo de software.

A continuación, se detallan las acciones realizadas en cada una de las etapas.

3.1. Análisis y especificación de requisitos

En primer lugar se analizaron las necesidades específicas del proyecto: existía un programa de Matlab que realizaba la selección de casos empleando MDA y se habían realizado pruebas en Python para la interpolación usando RBF con el algoritmo de Rippa [24] para la selección de variables, aunque con distintos parámetros.

Se decidió que el lenguaje de programación debía ser Python. En cuanto a si convenía más un entorno web, una aplicación para móvil o de escritorio, se optó por la primera opción por la diversidad de sistemas operativos convencionales (Windows, MacOSX y Linux) y de móviles (iOS y Android) que tenían los compañeros del proyecto.

Se procedió a crear un script-board con los elementos mínimos que debería incluir el

entorno (ver Anexo A: Scriptboard inicial), intentando reflejarlo con el máximo detalle en la web diseñada:

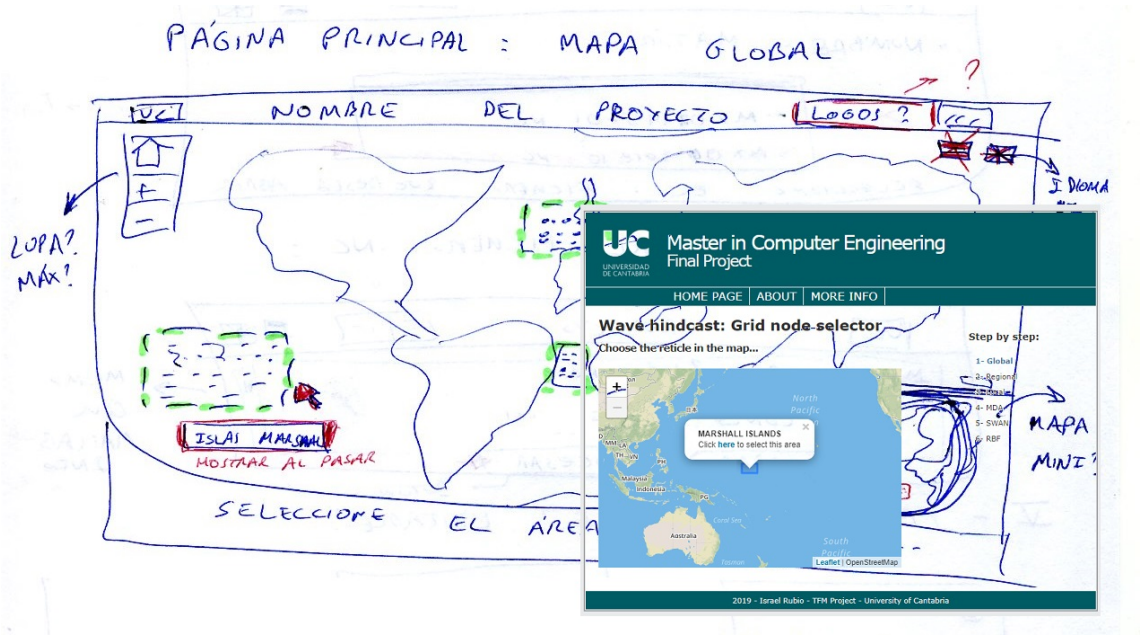


Figura 7: Scriptboard vs Diseño web final.

El diseño de requisitos fue más complicado: en un principio no se tenían claras todas las funcionalidades que debían incluirse. La mayor duda era si sería preciso un sistema de login para acceder a la web y qué diferencias de funcionalidad existirían entre el usuario y el administrador. Se decidió que no era necesaria la diferenciación de roles ni de un sistema de gestión de usuarios: todo usuario podría acceder al contenido y realizar el proceso completo sin ningún tipo de limitación.

A continuación, se enumeran los requisitos funcionales, no funcionales y casos de uso definidos:

a) Requisitos funcionales

Los requisitos funcionales son aquellos requisitos de alto nivel que debe satisfacer el sistema [23]. Dado que el proceso será lineal y, una vez seleccionados los parámetros nodo, casos y punto de propagación, el proceso sólo estará completo si se obtienen los resultados, sólo existe un requisito funcional:

ID	Descripción
RF-01	El usuario obtendrá la propagación del punto seleccionado para el nodo y número de casos elegidos y los resultados aparecerán de forma numérica (tabla de datos) y gráfica (representación).

Cuadro 1: Tabla de requisitos funcionales.

b) Requisitos no funcionales

Los requisitos no funcionales no son obligatorios, pero garantizan la calidad del producto. En el capítulo 6 de Ingeniería del software de Sommerville [27] se recoge que los requisitos no funcionales son de 3 tipos: de producto, de organización y externos.

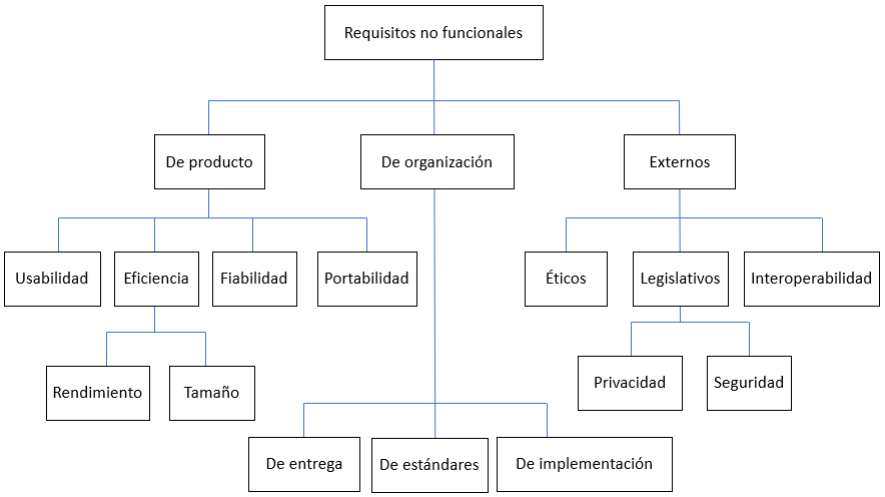


Figura 8: Esquema de requisitos no funcionales de Sommerville.

Sin embargo, si queremos cumplir con los estándares de calidad, debemos fijarnos en los requisitos no funcionales que vienen indicados por la norma ISO 25.010:

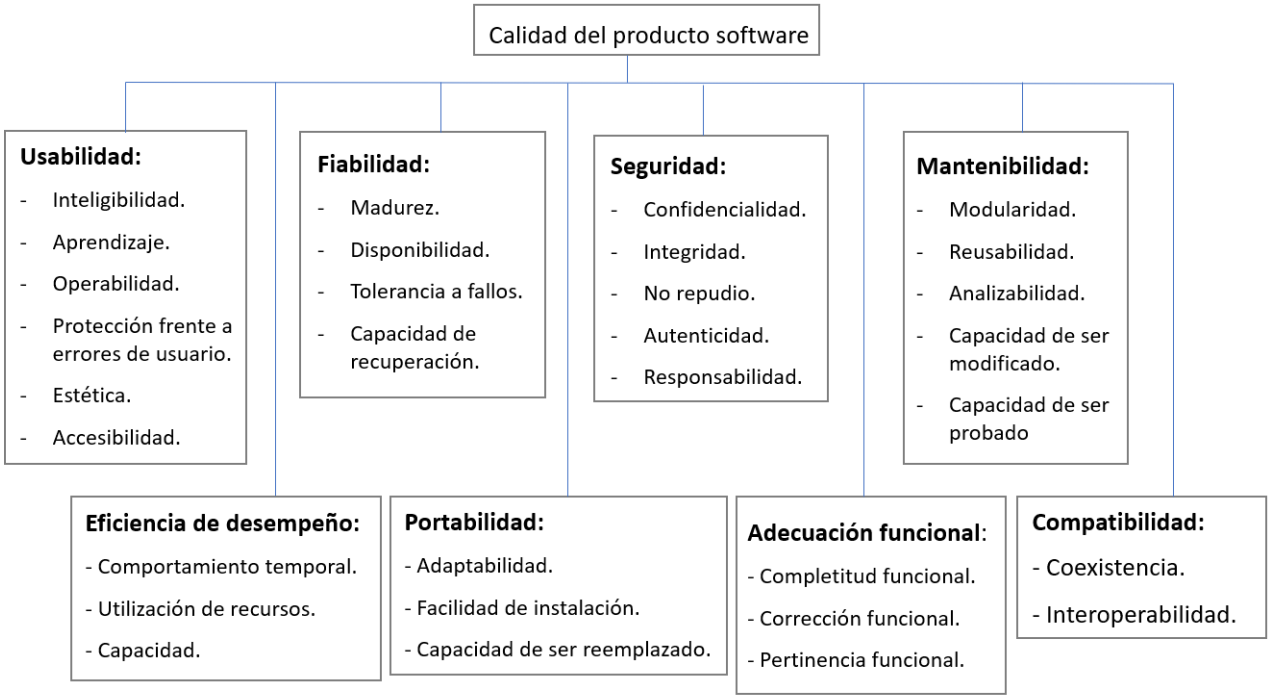


Figura 9: Esquema de requisitos no funcionales según la norma ISO 25.000.

En la siguiente tabla se recogen los requisitos no funcionales encontrados para nuestro sistema:

ID	Descripción
RN-01	La aplicación debe ser intuitiva, fácil de usar por cualquier usuario.
RN-02	El proceso debe hacerse en un tiempo aceptable y sin consumir excesivos recursos.
RN-03	Se precisa conexión a internet y un navegador de internet compatible para mostrar la web.
RN-04	El sistema debe poder ser mantenido de forma sencilla y adaptarse a nuevos requisitos y funcionalidades.
RN-05	En caso de error, el sistema debe mostrar el fallo y no alterar los datos almacenados.
RN-06	El sistema podrá recuperarse a un estado funcional en caso de fallo catastrófico.
RN-07	La web está maquetada empleando el estándar aceptado en HTML5.
RN-08	El contraste fondo/letra será adecuado para no cansar la vista del usuario.
RN-09	No existen puertas traseras que permitan un uso fuera del flujo lógico del sistema.
RN-10	El sistema debe ser portables en plataformas GNU/Linux, MacOS y Windows.

Cuadro 2: Tabla de requisitos no funcionales.

Como se puede apreciar, existe al menos un requisito no funcional para cada una de las competencias requeridas en la norma ISO 25.000:

- Usabilidad: requisitos RN-01 y RN-08.
- Fiabilidad: requisitos RN-05 y RN-06.
- Seguridad: requisito RN-09.
- Mantenibilidad: requisito RN-04.
- Eficiencia de desempeño: requisito RN-02.
- Portabilidad: requisito RN-10.
- Adecuación funcional: requisito RN-07.
- Compatibilidad: requisito RN-03.

c) Casos de uso

En la siguiente tabla se recoge el caso de uso implementado y que satisface el requisito funcional definido previamente:

Id + Nombre	1001 ObtenerPropagación.
Autor	Israel Rubio.
Fuente	Funcionalidades propuestas y observación etnográfica de la web: https://uoa-eresearch.github.io/storm_surge
Actor principal	Usuario.
Actores secundarios	
Descripción	El usuario completa el proceso de propagación en las coordenadas definidas.
Evento de activación	El usuario activa el evento ObtenerPropagación.
Precondición	
Garantías si éxito	El usuario completa el proceso.
Garantías mínimas	El sistema se mantiene consistente.
Escenario principal	1. El sistema recupera el formulario en forma de mapa. 2. El sistema muestra las retículas disponibles al usuario. 3. El usuario elige una retícula [1] en el mapa mostrado por el sistema. * 4. El sistema valida que los datos introducidos son correctos. * 5. El sistema recupera un mapa ampliado en función de los datos introducidos por el usuario. 6. El sistema muestra los nodos disponibles al usuario. 7. El usuario elige el nodo y el número de casos para analizar. * 8. El sistema envía los resultados y se realizan los cálculos. 9. El sistema muestra los resultados de forma gráfica y permite descargarlos *
Extensiones	1- No se declara actividad del usuario. 1.1 - Se cancela el caso de uso. 3- El usuario no ha introducido alguno de los datos obligatorios. 3.1- El sistema muestra un mensaje de error al usuario. 3.2- El sistema vuelve al paso anterior. 4- El caso elegido no es correcto [2]. 4.1- El sistema muestra un mensaje de error al usuario. 4.2- El sistema vuelve al paso anterior. 7- El sistema comprueba que hay datos. 7.1- El sistema muestra un mensaje de error 7.2- El sistema vuelve al paso anterior. 9- Se produce un problema con la base de datos. 9.1- El sistema muestra un mensaje de error al usuario. 9.2- El usuario notifica al sistema que ha leído el error. 9.3- El sistema cancela el caso de uso.
Comentarios	* Indica que existe una extensión para este escenario principal. [1] La retícula sólo será visible si el nivel de zoom lo permite. [2] Sólo se admiten números enteros entre 0 y el número de casos elegido.

Cuadro 3: Tabla de casos de uso.

3.2. Diseño

a) Diseño arquitectónico

Es común para realizar el diseño arquitectónico el empleo de un patrón de diseño para lograr de forma sencilla las funcionalidades deseadas y que nos proporcione unos atributos de calidad. De la gran variedad existente (patrón de capas, maestro-esclavo, de intermediario, de igual a igual, de pizarra, MVC,...) optamos por el patrón de modelo-vista-controlador: este patrón desacopla los componentes en capas de forma que el desarrollo se agiliza y es de gran uso en entornos web. Aunque en principio se planteó usar Django [9], por necesidades de funcionalidad se recurrió finalmente a Flask [1], más complejo y menos documentado, pero más flexible.

MVC [11] es un patrón que divide el proyecto en 3 partes:
- Modelo: parte que contiene las funcionalidades básicas.
- Vista: parte que controla lo que se muestra al usuario.
- Controlador: parte que controla las acciones del usuario.

b) Diseño detallado

El esquema del diseño inicial del entorno se muestra en el siguiente diagrama:

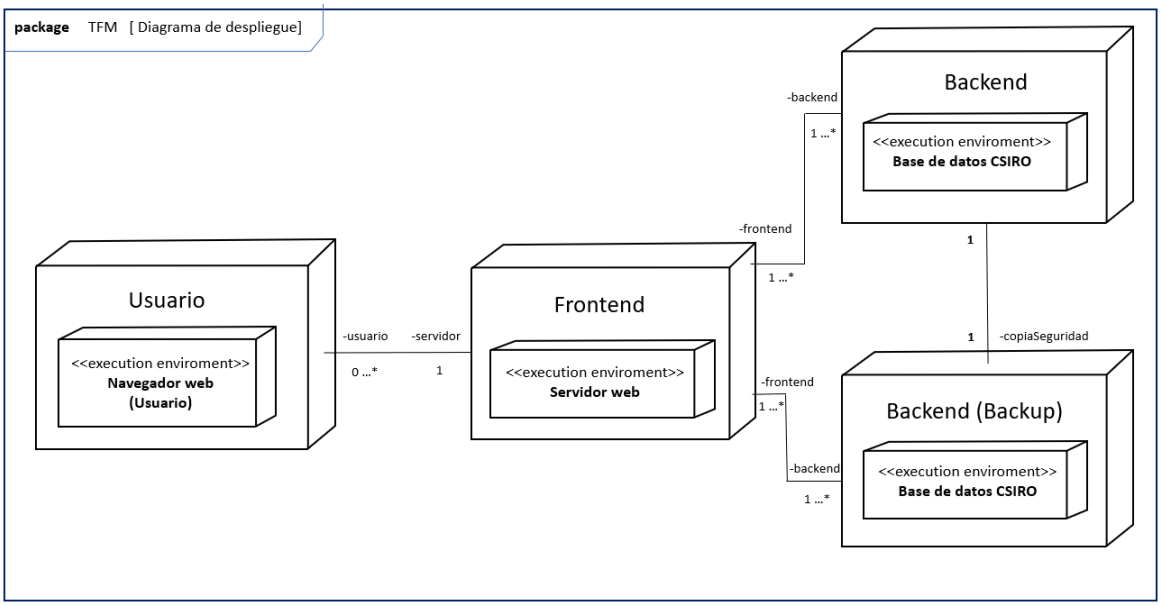


Figura 10: Diagrama de despliegue.

En principio no vamos a necesitar que se inicie sesión y se tendrá acceso completo a todas las funcionalidades por lo que no hacemos la distinción entre usuario y administrador en la parte del cliente. Cualquier usuario con un navegador compatible podrá realizar su consulta al servidor, cuyo front-end está basado en Python, y obtener los datos de nuevo en su navegador tras realizar las operaciones pertinentes. Por su parte, el back-end estará formado por la base de datos de CSIRO con los nodos de estudio de las Islas Marshall y existirá un servicio de back-up de toda la información que garantice que puede ser recuperada en caso de fallo.

c) Diseño desarrollado

Para el desarrollo de la web se optó por la metodología TDD [32]: se trata de un modelo de diseño ágil basado en el desarrollo guiado por pruebas. Este modelo es aplicable en este caso porque ya sabemos el resultado que queremos obtener (en el caso de MDA tenemos el archivo de Matlab y para RBF una versión en Python pero que usaba diferentes variables y método de validación).

Las ventajas principales de TDD son:

- Implementar las funciones justas que el cliente necesita.
- Minimizar el número de defectos que llegan al software en fase de producción, por lo que la calidad aumenta.
- Producir software modular, altamente reutilizable y fácil de modificar.

El ciclo de desarrollo, definido por Kent Beck y que se denomina comúnmente Red-Green-Refactor, tiene 3 etapas:

1. Elegir un requisito y probar que falla: se elige el requisito más fácilmente implementable, se escribe una prueba para dicho requisito, teniendo en cuenta las especificaciones y los requisitos de funcionalidad que está por implementar y se verifica que la prueba falla (RED). Si la prueba no falla es porque el requisito ya estaba implementado o porque la prueba es errónea.
2. Escribir la implementación: reescribir el código de manera más sencilla para hacer que la prueba funcione (metodología KISS) y comprobar que no da error (GREEN).
3. Eliminación de duplicados: el paso final es la refactorización para eliminar código duplicado. Se hacen pequeños cambios cada vez y luego se corren de nuevo las pruebas hasta que funcionen (REFACTOR).

Figura 11: Esquema de desarrollo TDD.

Como referencia para Python, se usaron las referencias: Head First in Python [2] y Python Data Science Handbook [34].

c.1- MDA

En el caso del MDA, se realizó primero en Matlab el proceso de obtención de los puntos para distinto número de casos (50, 100, 500 y 1.000) con un ejemplo que ya se sabía que funcionaba [7] y luego se programó un código en Python que replicara los pasos en la misma secuencia y se compararon los resultados.

El proceso tuvo las siguientes fases:

1. Prueba con Matlab Engine: como una primera aproximación al problema, se trató de acoplar el Matlab Engine en el proyecto. Este paquete para Python se incluye en Matlab, pero no se encuentra activado por defecto, por lo que es necesario instalarlo desde un terminal desde un ordenador que ya tenga instalado Python. Se probó a instalarlo en Python 2.7 y funcionó perfectamente y al primer intento, es extremadamente simple ya que sólo son 3 líneas de texto en Python: importar el Matlab.engine, iniciar Matlab en segundo plano y finalmente ejecutar el proceso (se puede consultar el código en el DVD adjunto, en la carpeta de Pruebas). Sin embargo, la prueba tenía una serie de desventajas evidentes:

   - Sólo funcionaba en versiones de Python inferiores a 3.6: aunque en principio se comenzó programando en Python v2.7, se decidió que la versión final debía estar en v3.7 ya que se espera que para final de año se deje de dar soporte a la versión 2 y el proyecto va a seguir usándose después de esa fecha.
   - Los tiempos de ejecución eran muy altos: en Matlab un fichero de 2.000 datos que se ejecuta con 100 casos tarda unos 10 segundos en completar MDA, con Python usando MatlabEngine tardaba cerca de un minuto.
   - Se necesita tener Matlab instalado para usar Matlab Engine: se pretende tener una versión autónoma y que sólo use software libre por lo que no es viable.

2. Prueba en R: dado que Matlab Engine no nos permitía obtener lo que queríamos, se propuso el emplear una versión de MDA escrita en R y ejecutar directamente usando la librería rpy2 de Python. Sin embargo, se producían los mismos problemas: aunque el procesado en R es rápido, llamar al proceso desde Python encarecía el proceso en excesivo tiempo de procesado y además se tenía que tener instalado el software de R, que en este caso sí es gratuito pero suponía obligar al usuario final a más requisitos técnicos.

3. Programación en Python: desechando también R, sólo quedaba reproducir totalmente el código de Matlab en Python. Aunque era complejo, se pudo portar casi línea a línea gracias al uso de la librería Numpy de Python. Tras varios intentos fallidos se consiguió que el resultado obtenido fuera casi idéntico al de Matlab (difería en el octavo decimal, así que es despreciable). La parte más compleja fue la variable circular θ, que debía ser normalizada de forma diferente al resto de variables puesto que 0 y 360 grados (máximo y mínimo de la variable) coinciden. Para la realización de las gráficas se empleó la librería Matplotlib [35], ajustando también colores y diseños para que salieran exactamente igual que en Matlab para poder compararlos. Los tiempos de ejecución con Python resultaron también muy similares a los de Matlab (el mismo fichero del caso anterior tardaba 10 segundos en Matlab y 8 segundos en Python).

4. Ajuste con datos reales: una vez probado que funciona correctamente para un conjunto de datos de prueba, se procede a usar un conjunto de datos real. Para que sea lo más similar posible, se toman 4 meses de datos medidos cada hora en la posición más cercana a Majuro (del 1 de enero al 30 de abril de 1979, 120 días, lo que supone un total de 2.880 mediciones). En el caso de demo trabajábamos con datos de periodo directamente, pero en los datos reales tenemos frecuencias por lo que tenemos que cambiar las variables para que haga el inverso del valor de frecuencia. La ejecución en Matlab es correcta pero al probar con Python no se obtiene el resultado correcto. Tras distintas pruebas, descargando de nuevo el fichero de entrada, cambiando de nombre las variables, releyendo toda la documentación sobre algoritmos en Python [14] y comprobando de nuevo el código línea a línea, se descubrió que el error provenía de la base de datos de CSIRO: el primer valor histórico que posee (1 de enero de 1979) es H=0, f=0, θ=0; al realizar la inversa de la frecuencia se obtiene infinito y MDA considera todas las distancias infinitas por lo que el proceso nunca termina pero tampoco advierte del error (Matlab considera el dato inicial como NaN y no operaba con él). Por tanto, se procede a cambiar las variables para que inicien desde el segundo dato, con lo que se consigue que funcione tanto en Matlab como en Python y con resultados idénticos.

```
# input: parameters
hs = xds_wvs.hs.values[1:]
hs = [np.float(j) for j in hs]
tp = 1 / xds_wvs.fp.values[1:]
tp = [np.float(k) for k in tp]
wd = xds_wvs.dir.values[1:]
wd = [np.float(l) for l in wd]
tiempo = xds_wvs.time.values[1:]
tiempo = [np.float(m) for m in tiempo]
```

Figura 12: Código de MDA en Python (eliminando primer valor).

5. Mejora de resultados: una vez completado el proceso, se pretende mejorar el tiempo de ejecución:

   - Se limpia el código de comentarios innecesarios, útiles a la hora de escribir pero que ya no sirven: en Python se puede comentar una línea usando el símbolo de almohadilla o comentar varias a la vez usando tres comillas simples al inicio y al final. Aunque pueda parecer trivial, no se lee todo lo que comienza por # pero sí lo incluido entre comillas (aunque no se ejecute).
   - En muchas expresiones matemáticas se recurre a divisiones o a llamar a la libreria de Numpy para usar π. Se procede a multiplicar por el inverso y sustituir todo lo posible por sus valores numéricos.

   Aunque puede parecer que no es mucho, sólo con estos cambios se consigue pasar de un tiempo de 8 segundos a menos de 1 segundo, por lo que se da por terminado aqui el proceso para MDA. Viendo el fichero original de Matlab frente al resultado final en Python, se puede observar que es prácticamente idéntico: por ejemplo, la extensión de Matlab *dist-normalizada* frente al bloque de función *normalized-distance* de Python:

```
function [qerr bmu] = dist_normalizada(M, D, PosEscalares, PosDireccionales)

    dMD(:,PosEscalares) = abs(D(:,PosEscalares)-M(:,PosEscalares));
    dMD(:,PosDireccionales) = min(abs(D(:,PosDireccionales)-M(:,PosDireccionales)),
       2*pi-abs(D(:,PosDireccionales)-M(:,PosDireccionales)))./pi;
    Dist = abs(sum(dMD.^2,2));
    [qerr bmu] = min(Dist);
```

```
def normalized_distance(m, d, ix_scalar, ix_directional):

    for ix in ix_scalar:
        dif[:, ix] = d[:, ix] - m[:, ix]
    for ix in ix_directional:
        ab = np.absolute(d[:, ix] - m[:, ix])
        dif[:, ix] = np.minimum(ab, 6.2831853072 - ab) * 0.3183098862
    dist = np.sum(dif**2, 1)
    return dist
```

Figura 13: Comparación del código de Matlab frente a librería en Python.

c.2- SWAN

Existía ya una versión de SWAN empleada para el proyecto ESTELA [21], por lo que sólo se tuvo que ajustar a las variables utilizadas y cambiar el formato del fichero de salida de MDA para que coincidiera con el de entrada de SWAN (el fichero creado por Matlab era .dat y se decidió mantenerlo para mostrar en la versión web, pero hubo que añadir también el guardado en formato .pkl, más común en Python. Como además este programa es sujeto de los trabajos finales de máster de varios miembros del departamento, no se modificó más que lo esencial para que las variables tuvieran el mismo nombre y se comprobó que funcionase.

c.3- RBF

Para RBF, al tener ya la referencia de resultados obtenidos en Matlab [6] y estar ya parte del código en Python, bastó con implementar la normalización para variables circulares, usar las mismas gráficas que en MDA pero con los nuevos datos de salida y se decidió cambiar el método de validación cruzada de leave-one a $k-$fold para que el proceso fuese más rápido. Al igual que para MDA, se limpió código y corrigieron expresiones numéricas, consiguiendo reducir el tiempo de procesado a unos 2 minutos.

El único problema se presentó en la selección del valor de k en la validación cruzada: se inició con $k = 4$ y daba error porque el número de casos no era divisible en partes enteras por 4. Al tomar $k = 5$ se aumentaba el tiempo de procesado pero se solucionó el problema anterior. En el DVD se incluye en la carpeta Pruebas el código original del que se partió para la realización de esta tarea.

c.4- Flask

Como se ha comentado anteriormente, para el desarrollo de la web se optó por un patrón arquitectónico MVC basado en el framework Flask, tomando como referencia Flask Blueprints [22] para la parte de programación en Python y Head First HTML5 Programming [10] para HTML, y usando PyCharm como IDE de Python.

En un principio se comenzó el proyecto usando Django ya que es el framework más común, con mayor documentación y mejor valorado para el desarrollo en Python, pero tuvo que desecharse porque, al intentar portar los códigos de los algoritmos matemáticos el proceso, entraba en bucle y no terminaba. En Flask, en cambio, siguiendo los pasos explicativos de creación de una aplicación web de las referencias mencionadas anteriormente, se pudo crear una primera versión que lanzaba el código de MDA y guardaba los resultados en la carpeta indicada. Las etapas que se realizaron fueron las siguientes:

1. Creación de un mapa global: se comenzó programando el mapa global en Python usando la extensión Basemap de Matplotlib, obteniéndose una representación cartográfica en proyección de Mercator que además podía ser manipulado para hacer zoom. El código realizado se incluye en el DVD en la carpeta de Pruebas. Sin embargo, tenía varios inconvenientes:

   - La extensión Basemap es muy antigua, funcionaba bien en Python v2.7 pero en v3.7 no se instalaba.
   - El tiempo de procesado era demasiado largo, se tardaba casi 2 minutos en cargar el mapa completo.
   - La imagen tenía poca resolución, si se aumentaba mucho el zoom se pixelaba y la imagen de satélite dejaba muchas zonas en negro.

   Por todas estas razones, se buscó la alternativa de un mapa en JS que admitiera importar archivos NetCDf4 y se encontró como solución usar Leaflet, una libreria de código abierto que se puede embeber fácilmente en HTML y que permitió crear una representación cartográfica global centrada en la región de estudio (Islas Marshall), pudiendo añadirse en un futuro otras regiones en función de las necesidades de los proyectos. Se siguieron las especificaciones de su página web para la inclusión en nuestro código.

Figura 14: Primera versión de la página de inicio (mapa global).

2. Selector de puntos: una vez elegida la región de estudio, se debía realizar un mapa de profundidad con la zona ampliada y que mostrase los puntos de estudio, permitiese elegirlos y, además, escoger el número de casos de estudio. El dibujo del mapa se realizó en Python con Matplotlib cargando de momento sólo 3 puntos de estudio, cada uno con distinto rango de tiempos para poder evaluar luego los tiempos de carga y procesado al aumentar la carga de trabajo. En una primera versión se pretendía añadir una página que permitiera descargar los datos de CSIRO desde la web pero el hecho de que precise registrarse y que los tiempos de descarga varían entre 30 minutos y varias horas en función del número de datos hizo que se desechase la idea y se cargasen ya archivos predescargados. Los tres archivos de puntos utilizados fueron:

    - n0: se trata del mismo archivo empleado en las pruebas de MDA, es un punto en la zona oeste de Majuro y contiene 4 meses de datos medidos cada hora desde el 1 de enero al 30 de abril de 1979, es decir, 2,880 datos.
    - n1: al descargar los datos completos desde 1979 hasta 2018 se produjo un fallo ya que, desde 2013, los nombres de las variables habían cambiado. Por ello se decidió tomar el segundo punto desde el 1 de enero de 2013 al 31 de diciembre de 2018, es decir, 52,584 datos.
    - n2: el último punto se tomó con el máximo valor posible de datos, desde el 1 de enero de 1979 hasta el 31 de diciembre de 2012, es decir, 298,032 datos.

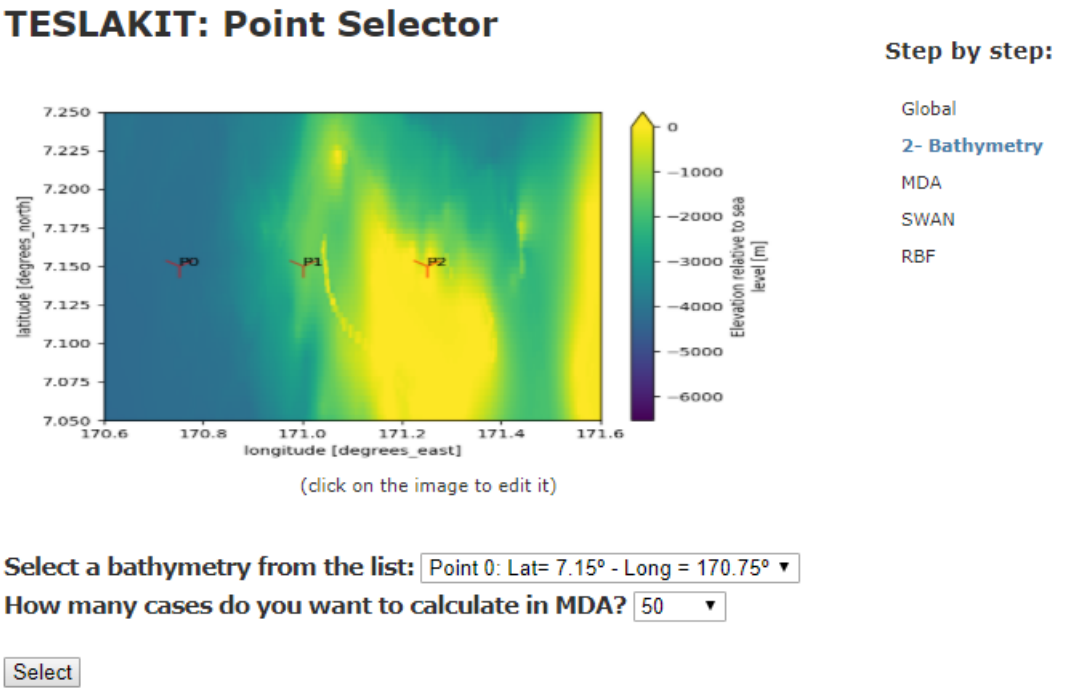


Figura 15: Primera versión de la página de selección de puntos.

Se ha incluido en el DVD en la carpeta Pruebas una copia del código en Python que se preveía integrar para realizar la descarga de los datos de CSIRO.

3. MDA: en este punto la parte web era sencilla pues sólo se tenía que mostrar los datos obtenidos con el código de Python programado para el MDA.

En Django, ésta era la parte donde no se consiguió continuar porque el servicio se detenía y se optó por usar Flask, donde funcionaba sin problemas.

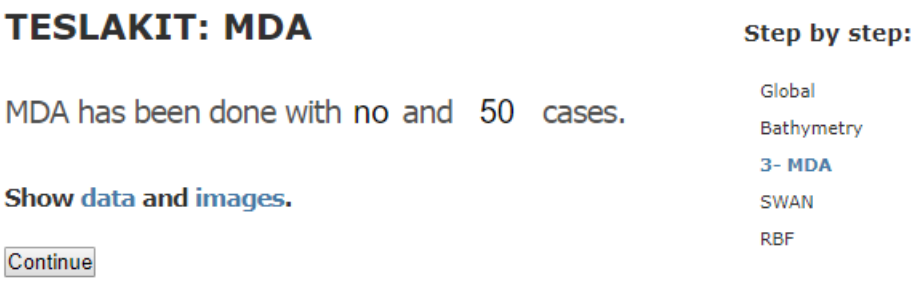


Figura 16: Primera versión de la página de MDA.

4. SWAN: como el programa ejecutaba todo en el lado del servidor, esta página era meramente informativa y para tener ordenado el proceso.

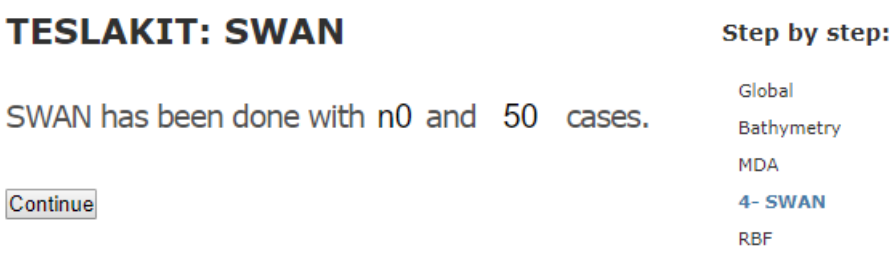


Figura 17: Primera versión de la página de SWAN.

5. RBF: al igual que en MDA, sólo se debía mostrar el resultado como fichero de datos y la gráfica. Se añadió además un botón para que terminara el proceso y volviera a la página de inicio.

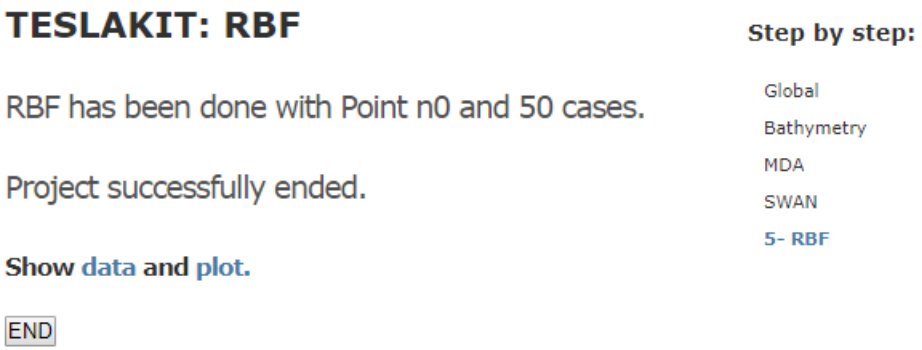


Figura 18: Primera versión de la página de RBF.

6. Cabeceras y CSS: se decidió añadir una página con la información del proyecto TESLAKIT, al que pertenece el estudio de condiciones de Majuro. Además, se ajustó la cabecera para que usara el logotipo de la Universidad de Cantabria, con los colores institucionales, y se ajustaron las hojas de estilo CSS para todas las páginas creadas.

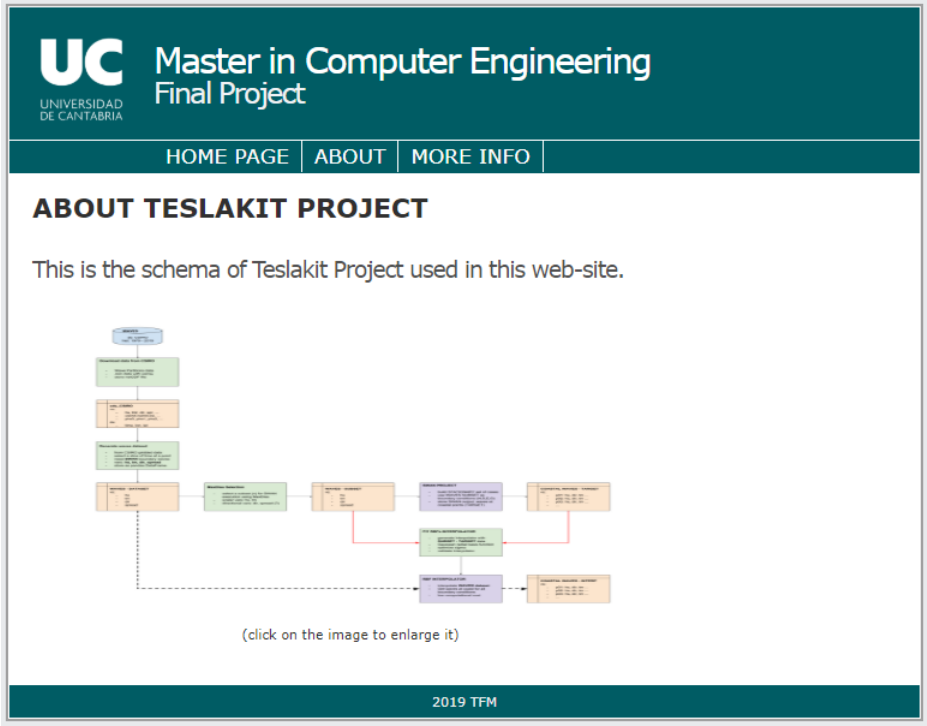


Figura 19: Primera versión de la página de información del proyecto con cabecera.

3.3. Implementación

Tras completar el diseño y obtener una versión funcional, se procede a su despliegue:

NAS

Al llegar al Departamento de ingeniería se constató que se estaban utilizando 4 cuentas premium de Dropbox de 1TB cada una para trabajo del día a día y que se disponía de 3 discos duros externos de 4 TB cada uno para almacenar las bases de datos descargadas. Sin embargo, mucha de la información estaba repetida, no existían copias de seguridad y con la llegada de los nuevos integrantes se buscaba una solución que fuera rentable y eficaz. Se propuso entonces la posibilidad de cancelar los servicios en la nube y comprar un servidor NAS para el Departamento, siempre y cuando fuera posible (por razones de contratación administrativa). Tras contactar con el Servicio de Informática de la universidad, que corroboraron que estos dispositivos son admitidos, el Departamento de Matemáticas aconsejó el mismo modelo que utilizaban ellos: el diskstation DS418j de Synology, un servidor NAS de 4 bahías diseñado para que usuarios del ámbito privado y doméstico puedan administrar, proteger y compartir datos de manera eficaz.

Figura 20: Diskstation DS418J.

El DS418J posee un procesador de doble núcleo de 64 bits y la capacidad de cada volumen individual puede ser hasta 40 TB (se incluye copia del manual con toda las características técnicas en el DVD [29]), soportando 4 discos. En un principio, dado el presupuesto de que se disponía, se optó por adquirir el equipo y dos discos de 8 TB, a la espera de que en el ejercicio siguiente se pudieran adquirir otros dos discos de 8 TB, con lo que se tendría 16 Tb para uso completo y otros 16 TB para copia de seguridad. Sin embargo, antes de acabar el ejercicio 2018 se contaba con fondos remanentes y se optó por adquirir dos discos de 4 TB cada uno, dejando entonces el sistema en 12 TB para uso normal y otros 12 TB para backup. El uso por disco y su configuración se encuentra recogido en el apéndice C: Distribución del servidor de este trabajo.

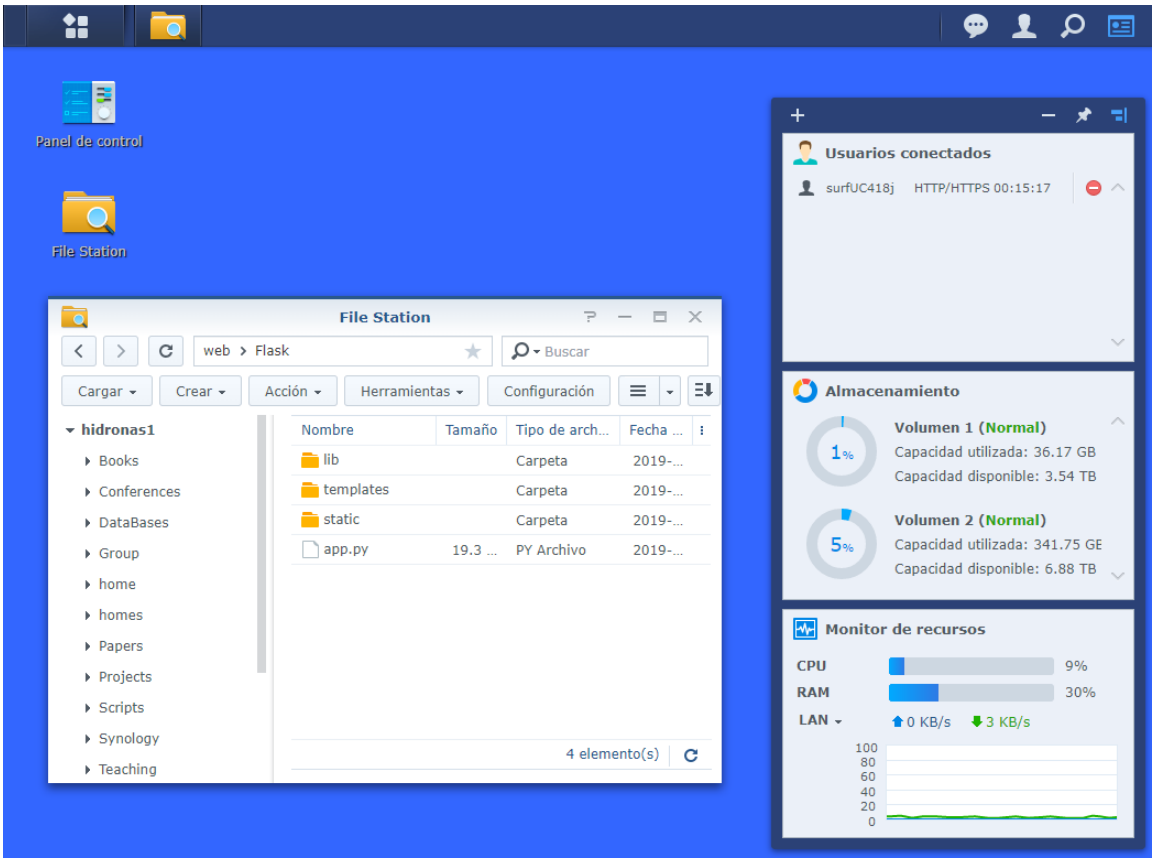


Figura 21: Escritorio de File station.

Se realizó la instalación del sistema operativo del fabricante (basado en Linux) y se pidió al Servicio de Informática de la Universidad de Cantabria que se diese de alta para su conexión a la red interna y acceso a internet. Se dio de alta y se conectó a la red interna pero no se dio acceso a internet porque, tras el ataque hacker PewDiePie a impresoras de 2018, se ha prohibido el acceso de todo periférico a internet. Por tanto, se tuvo que buscar una alternativa para poder acceder en remoto al NAS y se optó por la conexión Synology Quick Connect, que funciona a modo de túnel virtual, siguiendo las instrucciones del fabricante (se incluye copia del documento técnico en el DVD [28]).

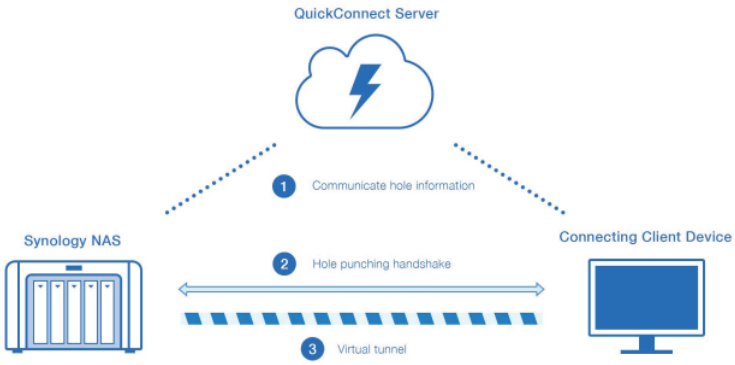


Figura 22: Synology Quick Connect.

Para terminar, se creó cuentas de administrador para el responsable del proyecto y de usuario para el resto de integrantes del grupo, se diseñó una página de inicio personalizada y se añadió una dirección IP estática: http://quickconnect.to/surfUC418

Figura 23: Página de acceso al NAS.

RAID

Como se ha comentado en el epígrafe anterior, como medida de seguridad adicional se propuso crear un RAID [33] con el nuevo servidor para que se garantizase una mayor integridad, tolerancia frente a fallos y tasa de transferencia estable. Synology admite 4 tipos de despliegue RAID en sus equipos:

- RAID 0: aunque no es un sistema RAID en sí porque no proporciona redundancia de datos, se considera como un formato de despliegue al ofrecer stripping: los datos se dividen en bloques entre las unidades disponibles para mejorar el rendimiento al permitir acceso concurrente a distintos datos. Precisa como mínimo 2 discos. La capacidad total del sistema es igual a la suma de todos los discos.

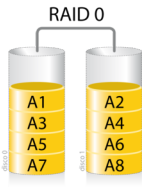


Figura 24: RAID 0.

- RAID 1: escribe datos idénticos en todas las unidades simultáneamente, proporcionando redundancia de datos (incluso es posible la replicación en más de un disco para aumentar el número de averías tolerables). Precisa 2 discos como mínimo. La capacidad total del sistema es igual al tamaño de uno de los discos.

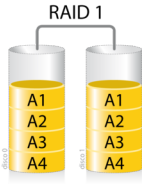


Figura 25: RAID 1.

- RAID 5: utiliza striping a nivel de bloque con paridad de datos distribuidos por todas las unidades integrantes, lo que proporciona una redundancia de datos más eficiente que RAID 1. Precisa de como mínimo 3 discos. La capacidad total del sistema es igual N-1 por el tamaño del disco, siendo N el número de discos.

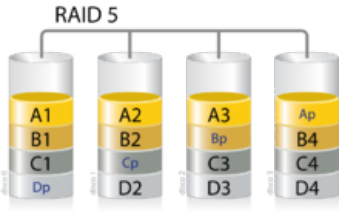


Figura 26: RAID 5.

- RAID 6: utiliza la paridad de dos capas de datos para almacenar datos redundantes en un espacio igual al tamaño de dos unidades, proporcionando un mayor grado de redundancia de datos que RAID 5. Precisa de como mínimo 4 discos. La capacidad total del sistema es igual N-2 por el tamaño del disco, siendo N el número de discos.

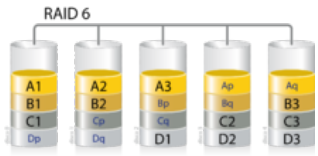


Figura 27: RAID 6.

- RAID 10: proporciona el rendimiento de RAID 0 y el nivel de protección de datos de RAID 1, mediante la combinación de unidades en grupos de dos, en los que se copian los datos. Precisa como mínimo de 4 discos y siempre en número par. La capacidad total del sistema es igual N/2 por el tamaño del disco, siendo N el número de discos.

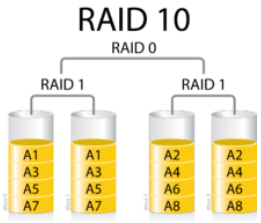


Figura 28: RAID 10.

Dadas estas condiciones y como teníamos 4 discos pero de distintas capacidades, para no perder espacio de disco las únicas opciones viables eran el RAID 0 y RAID 1. Se optó por crear dos volúmenes independientes y en cada uno de ellos crear un RAID 1, tal y como se muestra en la siguiente imagen:

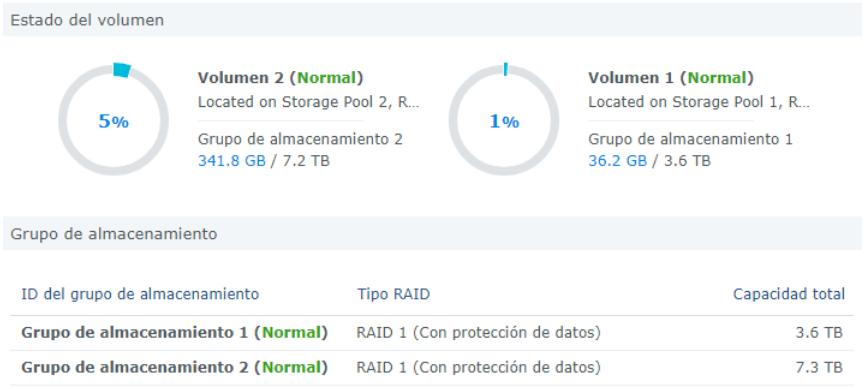


Figura 29: Grupos de almacenamiento del NAS.

Despliegue

Para el despliegue de la aplicación se sigue el procedimiento detallado en la referencia Mastering Flask Web Development [12]:

- A través del terminal se crea un entorno virtual en Python, en versión v3.7, donde se instalan los módulos necesarios: Flask, Pandas, Numpy, Matplotlib y Xarray. Esto permite portar el proyecto a cualquier ordenador que posea Python instalado (sin importar que sea una versión anterior).

- A continuación, se crea la aplicación web: app.py, añadiendo una ruta para cada una de las páginas web creadas anteriormente y definiendo los métodos GET y POST necesarios para el uso de datos.

- Para el correcto funcionamiento de Flask, es necesario un orden específico de los archivos en ciertas carpetas: las librerias de Python con los algoritmos matemáticos deben ir en la carpeta *lib*, las páginas web en formato HTML en la carpeta *templates* y todos los demás elementos (bases de datos, imágenes, hojas de estilo, código JS y el programa SWAN) en la carpeta *static*.

- Tras distintas pruebas, se consigue el despliegue de la aplicación en localhost (127.0.0.1), pudiendo abrirse desde cualquier navegador a través del puerto 5000. Adicionalmente, se sube una copia al servicio de internet Pythonanywhere para que el resto de personal del departamento pueda probarlo sin tener que instalar todo. Esta versión, limitada para evitar tiempos excesivos de procesado, puede ser consultada en: http://irl167.pythonanywhere.com

3.4. Pruebas

Una vez disponible la web, se pidió a los miembros del proyecto que actuaran como grupo de pruebas. En función de los ordenadores disponibles, se hicieron puebas en 3 sistemas operativos distintos (Windows, Linux Ubuntu y MacOs X) y con 5 navegadores web (Microsoft Edge exclusivamente en Windows, Apple Safari exclusivamente en MacOSX, Mozilla Firefox en Linux y Windows y Google Chrome en los 3 sistemas).

Los mejores resultados se obtuvieron con el PC de sobremesa (Hidronas5) ya que poseía una partición de Windows y otra de Linux, lo que permitió comparar resultados: al usar Firefox en ambos sistemas, los tiempos de procesado fueron prácticamente idénticos y respondía de forma satisfactoria con todos los puntos y casos.

Como única salvedad, en MacOSX se produjo un fallo con Safari en la visualización del mapa de Leaflet en la página principal, producido porque tenía una versión antigua de Javascript instalada y, por lo tanto, no atribuible al proyecto.

Se imprimieron todas las páginas web y se anotaron los comentarios, fallos y posibilidades de mejora. En el anexo B: Correcciones después de pruebas, se recoge copia de estos documentos.

3.5. Mejoras

Tras realizar el proceso completo, se somete el sistema a una serie de mejoras según las aportaciones del grupo de pruebas. A continuación, se indican los cambios realizados:

1. Cambios en Global: se decide que no aparezca el término TESLAKIT y se redefine la zona como Islas Marshall para, con vistas al futuro, añadir también luego Roi namur y otras islas al proyecto.

2. Cambios en Bathymetry: no convence la representación porque no se define la línea de costa. Se decide hacer el mapa en Leaflet pero con vista satelital y añadir todos los nodos existentes (aunque luego sólo permita trabajar con los 3 nodos de estudio principales). Se cambia el nombre de la sección a Regional y la selección de casos se hará en otra página.

3. Añadir Local: se debe crear una nueva página con la vista Local, en la página Regional debe dejar elegir el tamaño de separación de puntos de la rejilla y representarla aqui.

4. Transiciones: el hecho de que varios procesos tarden cierto tiempo en procesarse hace que sea necesario una animación durante la transición para indicar que se encuentra trabajando. Se decidió emplear una biblioteca de JS llamada Pace, ajustando los colores a los de la página web y con una animación en la esquina derecha que además degrada el color de la página web para indicar al usuario que no precisa su interacción hasta que finalice.

5. Cambios en Gráficas: se decide que la disposición de las gráficas de dispersión sea triangular en lugar de vertical y añadir también una vista temporal de cada variable. Además, se debe cambiar el color para mayor resalte.

6. Cambios en fichero de datos MDA: se debe incluir título, nombre de la variable y unidades de cada una de ellas. En principio se mostraba también la dispersión pero no se va a usar este valor de momento por lo que se deshecha.

7. Cambios en MDA y RBF: no convece el tamaño cambiante de la ventana, por lo que se decide que todas las páginas tengan las mismas dimensiones. Se debe añadir una vista previa de las gráficas y tabla de datos.

8. Cambios en SWAN: es la parte que más modificaciones tiene, se dividió en 2 fases:

   - Primero se seleccionan los puntos cercanos a la costa, se crean los casos SWAN, se muestran en pantalla y se ejecutan, mostrando a la vez una ventana con el caso sobre el que se itera.
   - Después, permite elegir el caso SWAN y dibujar una gráfica de dirección de viento y altura de ola.

   Fueron las tareas más complicadas, tanto por la complejidad de que el eje de coordenadas considera al Norte como origen de coordenadas (0 grados) y que evoluciona en sentido horario, siendo necesario emplear el opuesto del ángulo dado y sumar $\pi/2$, como por la selección de colores de la altura, que se decidió que fuera de blanco a negro con amarillo, naranja y rojo intermedios.

Se realizó un búsqueda en toda la gama de mapas de color hasta descubrir el adecuado: hot-reverse.

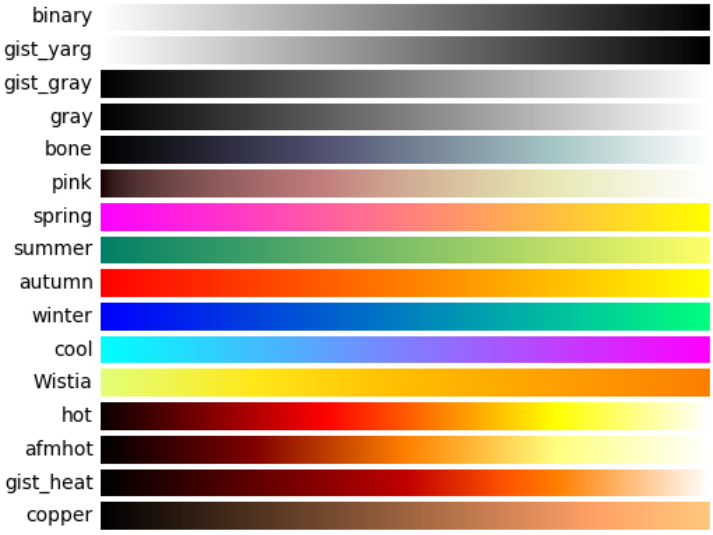


Figura 30: Colormap.

3.6. Análisis de calidad

La calidad del proyecto se analiza en cada una de las iteraciones empleando el software profesional Sonarqube para descubrir bugs, vulnerabilidades o fallos en el código y la deuda técnica asociada (número de horas necesarias para resolver todos los fallos):

Primera iteración

Una vez terminado el primer prototipo se procede a escanear la carpeta del proyecto completo, obteniéndose inicialmente los siguientes resultados:

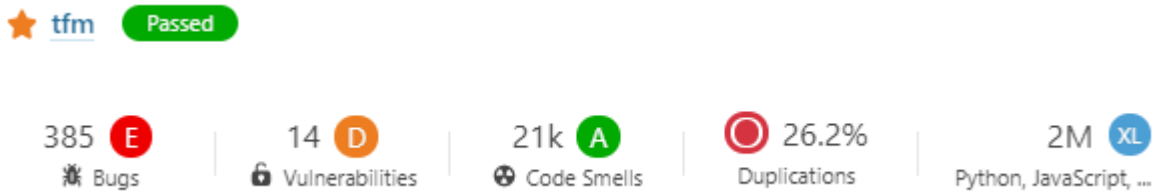


Figura 31: Resultados de Sonarqube del primer prototipo.

- 385 bugs, de los cuales 19 son de nivel E: bloqueante, 339 de *nivelC*: importante y 27 de *nivelB*: fallo menor.
- 14 vulnerabilidades, de las cuales una es de *nivelD*: crítico y 13 de *nivelB*: fallo menor.

- 21 mil líneas de código con algún tipo de problema, no revisten apenas gravedad por lo que son de *nivelA*: normal, pero la deuda técnica asciende a 360 días.

Sin embargo, al analizar con más detalle los bugs y vulnerabilidades observamos que los fallos se encuentran en la carpeta del entorno virtual, no en nuestro código. Volvemos a repetir el análisis, sin incluir esta vez a la carpeta del entorno virtual, y obtenemos:

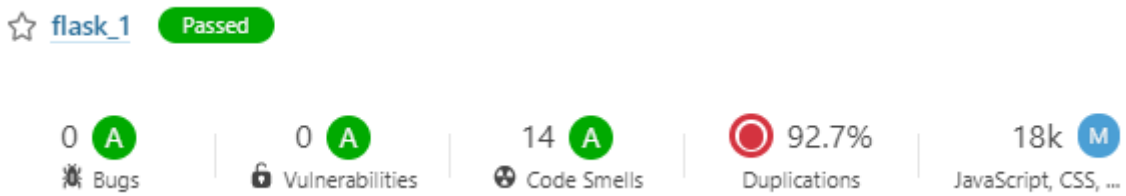


Figura 32: Resultados de Sonarqube del primer prototipo sin virtualenv.

Los bugs y vulnerabilidades han desaparecido, apenas hay 14 líneas de código con problemas y sólo se observa problemas en la duplicación de código, lo cual es comprensible porque el modelo SWAN crea una carpeta con archivos de input, output, profundidad y estado por cada caso, es decir, 1,750 carpetas por cada nodo. Estos resultados son bastante buenos y la deuda técnica se ha reducido a apenas 2 horas por lo que no realizamos ningún cambio más.

Segunda iteración

Tras modificar el proyecto siguiendo las indicaciones del grupo de pruebas, se procede de nuevo a escanear la carpeta del proyecto, eliminando esta vez la carpeta del entorno virtual, obteniéndose los siguientes resultados:

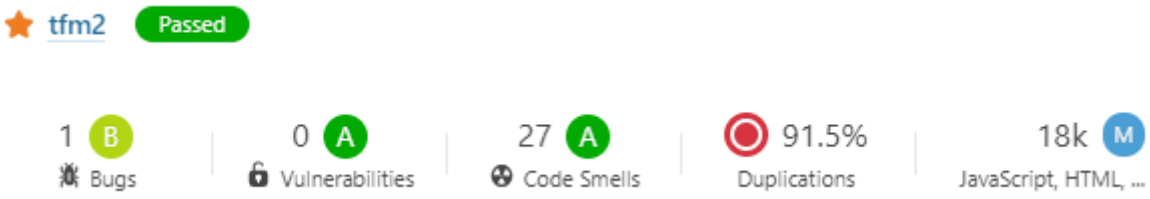


Figura 33: Resultados de Sonarqube del segundo prototipo.

Los resultados indican la presencia de un bug nuevo de *nivelB*: fallo menor. Al recabar información comprobamos que es debido al nuevo fichero creado: regional.html, al no haber incluido ningún título en el iframe empleado para el mapa de Leaflet.

Procedemos a añadir el atributo *title* y dejamos para el final la resolución de las líneas de código con problemas y los duplicados. Sí que se observa que ahora la deuda técnica ha subido de 2 a 6 horas, lo que nos recuerda que dejar un problema sin resolver hace que al avanzar en el proyecto se vuelva más complicado de solucionar.

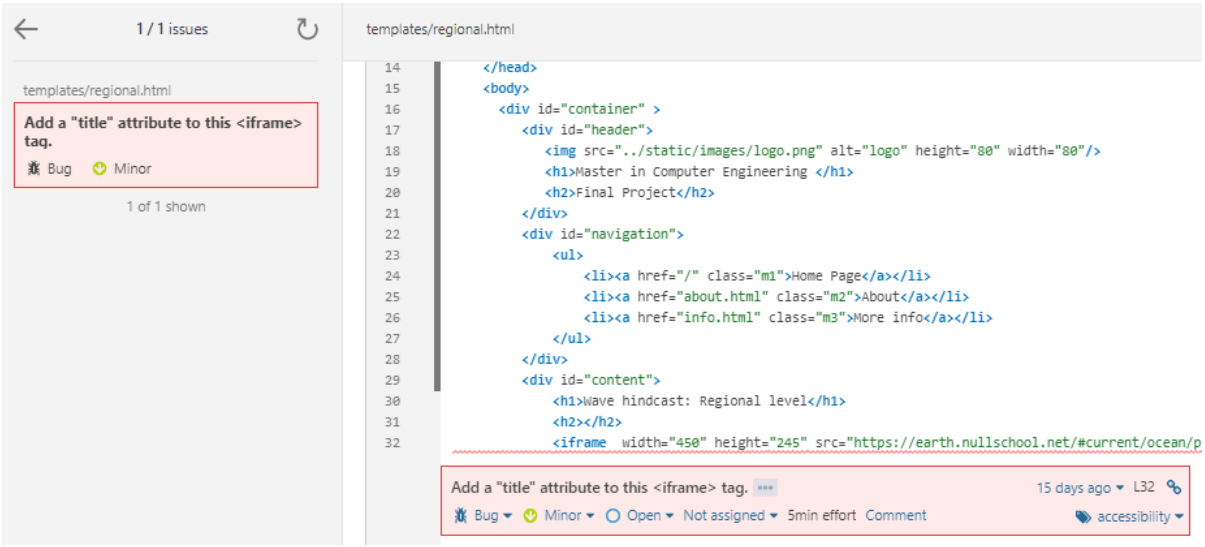


Figura 34: Bug del segundo prototipo.

Proyecto final

Una vez mostrado el proyecto final a los miembros del proyecto y viendo que se cumplen todas las expectativas, sólo queda corregir los fallos pendientes de líneas de código y duplicados:

- Los principales problemas de código vienen de la función definida para el seguimiento de casos en SWAN, por lo que procedemos a renombrarlo con el caso específico.
- Se descubre también que en las bibliotecas de MDA y SWAN existían comentarios superfluos por lo que se eliminan.
- En cuanto al código duplicado, se opta por eliminar la carpeta data para el análisis de Sonarqube y se observa que existe un archivo de Javascript de Leaflet repetido, por lo que se elimina del proyecto.

Una vez corregidos estos problemas, se escanea de nuevo el proyecto final, obteniéndose que todos los fallos han sido corregidos y no ya no hay bugs ni vulnerabilidades ni código duplicado:

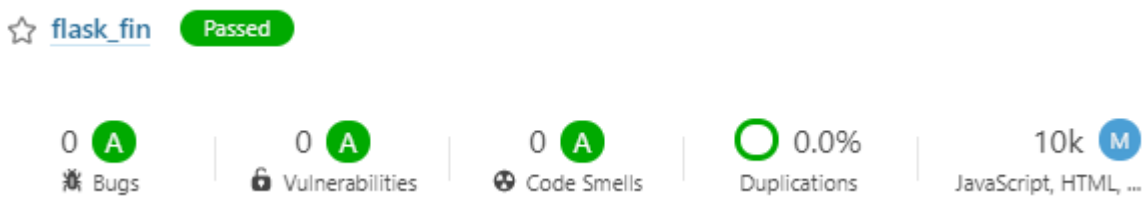


Figura 35: Resultados de Sonarqube del proyecto final.

3.7. Seguridad y Backup

Dada la preocupación actual por la seguridad de la información, se decidió instalar en el NAS un módulo de seguridad adicional para que, a parte del sistema RAID que hace copia de toda la información, alertara de cualquier comportamiento sospechoso o ataque por internet.

Figura 36: Módulo de seguridad del NAS.

Además, el sistema está programado para enviar un email al administrador cada vez que haya una incidencia y, una vez al mes, se crea un informe con toda la información relevante, incluyendo la degradación de los discos.

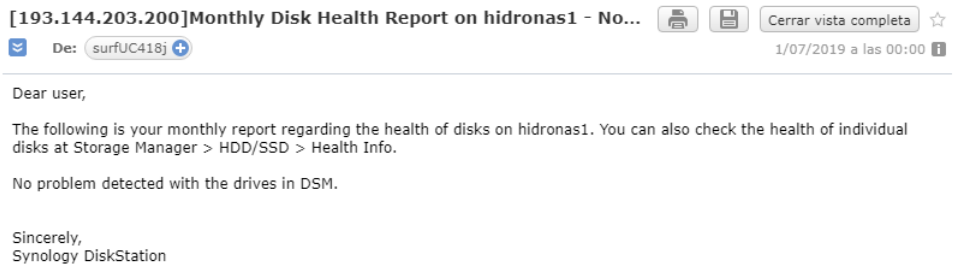


Figura 37: Informe de seguridad mensual.

Indicar además que se comprobó que el sistema RAID funciona correctamente: al mes de tener instalado el NAS se produjo un fallo de fábrica por sectores defectuosos en uno de los discos de 8 TB por lo que tuvo que ser reemplazado pero no se perdió información. Como medida extraordinaria, se decidió usar los discos duros externos que ya existían en el proyecto para realizar una copia completa cada 3 meses y tener así un backup en un despacho diferente a donde se encuentra el servidor (buena práctica recomendada por la norma ISO 27.001 y COBIT).

Capítulo 4

Conclusiones

4.1. Demostración

A continuación, se muestra el proceso completo de obtención de los datos de propagación, mostrándose lo que pasa en el lado del servidor y del usuario:

- Inicio: se inicializa la web desde el entorno virtual ejecutando Flask:

```
FLASK_ENV = development
FLASK_DEBUG = 0
In folder C:/Users/I/Documents/Flask
C:\Users\I\Documents\Flask\venv\Scripts\python.exe -m flask run
 * Environment: development
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figura 38: Ejecución de Flask.

Entrando en un navegador web, se obtiene la página de inicio con el mapa global:

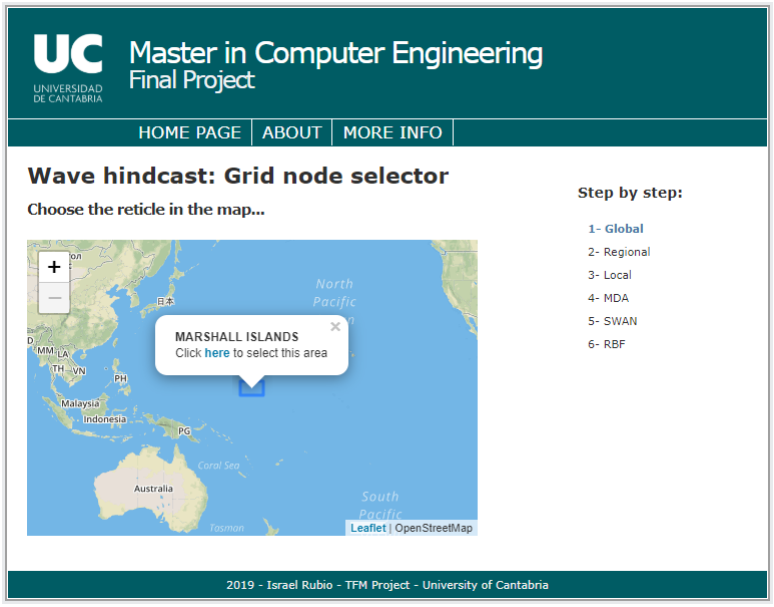


Figura 39: Página de inicio.

- Nivel regional: al hacer click sobre una de las regiones disponibles, se carga la vista regional:

```
Starting...
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/css/index.css HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/css/leaflet.css HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/js/script.js HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/js/pace.js HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/images/logo.png HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/js/jquery-3.3.1.min.js HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:09] "GET /static/js/leaflet.js HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:10] "GET /static/images/favicon.ico HTTP/1.1" 200 -
Regional map done
127.0.0.1 - - [04/Jul/2019 12:33:44] "GET /regional.html HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:45] "GET /static/images/marker-icon.png HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:33:45] "GET /static/images/marker-shadow.png HTTP/1.1" 200 -
```

Figura 40: Carga de datos a nivel del mapa regional.

En el navegador web, se obtiene el mapa regional con los nodos disponibles:

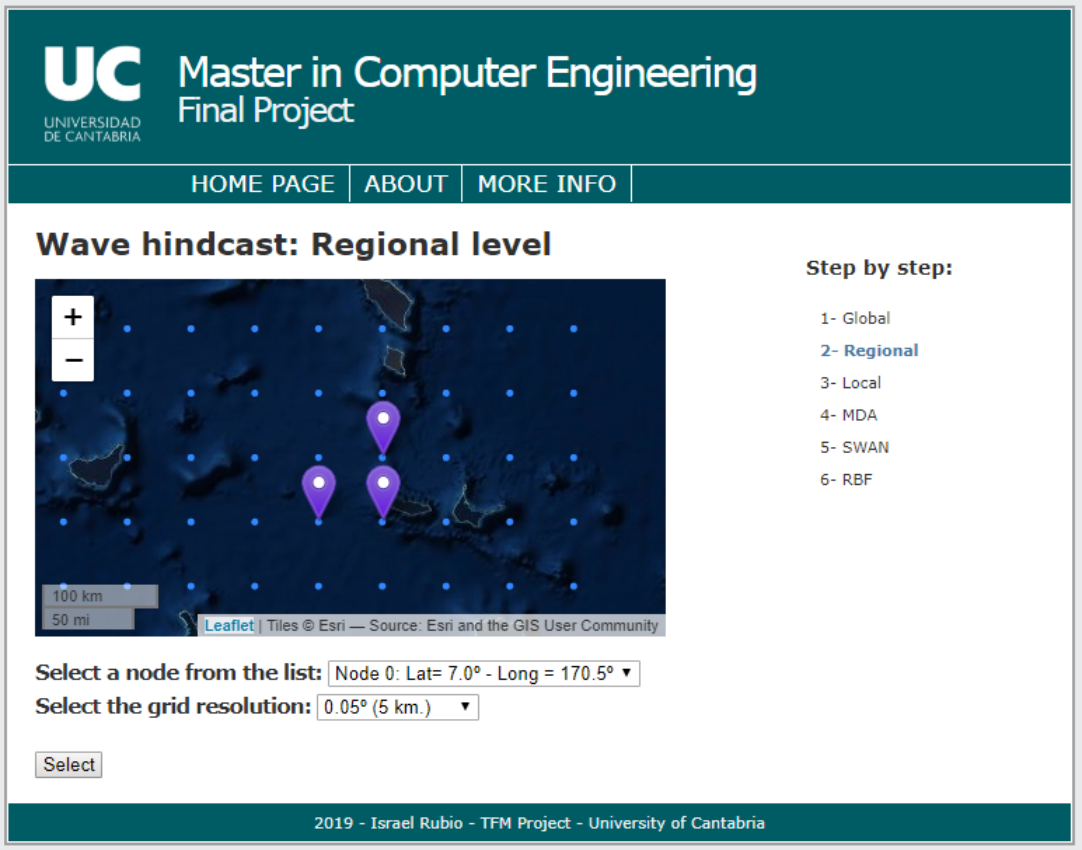


Figura 41: Mapa regional.

- Nivel local: al elegir el nodo n0 y 5 km de resolución, se carga la vista local y se muestra la latitud y longitud del nodo elegido:

```
Local map at ( 170.5 , 7.0 ) with grid 0.05 and node n0
127.0.0.1 - - [04/Jul/2019 12:35:17] "POST /local.html HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:35:17] "GET /static/images/n0/0.05.png HTTP/1.1" 200 -
```

Figura 42: Carga de datos a nivel del mapa local.

En el navegador web, se obtiene el mapa local con los nodos disponibles:

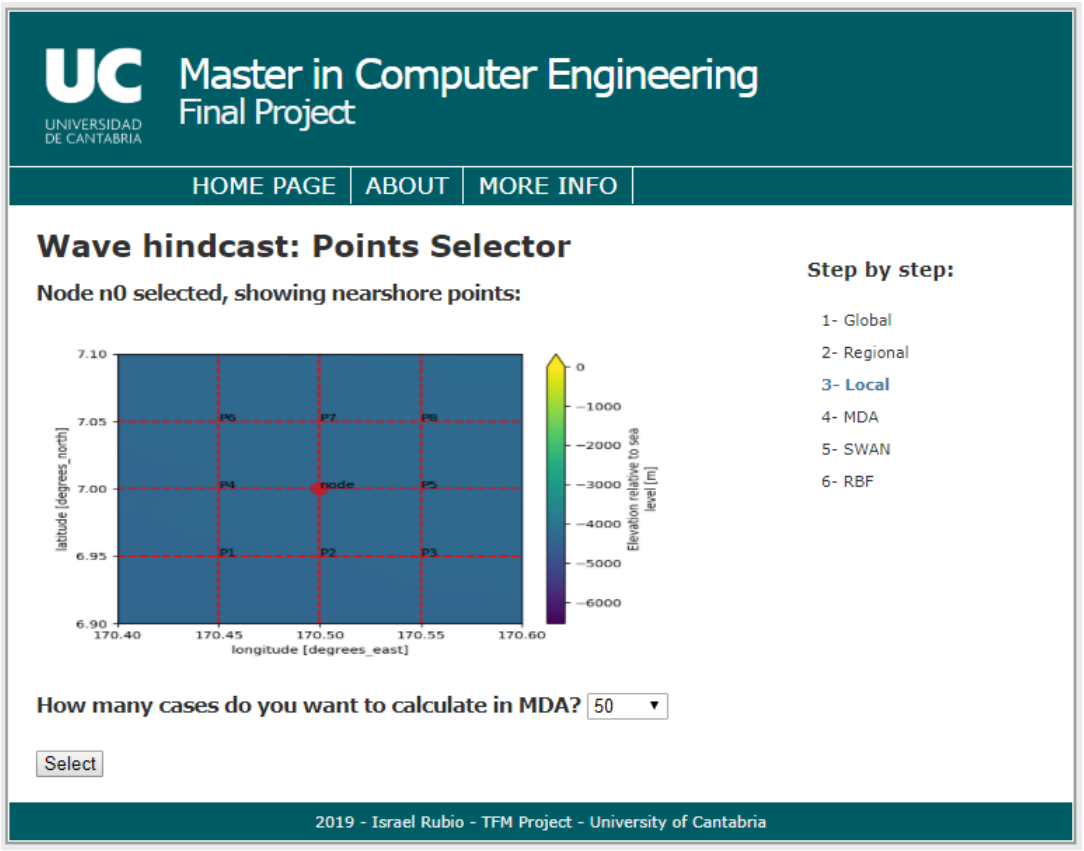


Figura 43: Mapa regional.

- MDA: al elegir el 50 casos de ejecución, el servidor comienza a ejecutar el algoritmo MDA. Se incluye el tiempo de inicio y fin para poder luego comparar los procesos por nodo y caso:

```
MDA in n0 with 50 cases
Start MDA: 2019-07-04 12:36:08.020263
End MDA: 2019-07-04 12:36:09.284887
127.0.0.1 - - [04/Jul/2019 12:36:09] "POST /mda.html HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:36:09] "GET /static/data/n0/50/MaxDiss1.png HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:36:09] "GET /static/data/n0/50/MaxDiss2.png HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:36:09] "GET /static/images/data.png HTTP/1.1" 200 -
```

Figura 44: Ejecución del MDA.

En el navegador web, comienza una transición mientras se hacen los cálculos y, al terminar se muestra un mensaje de que el proceso se ha completado:

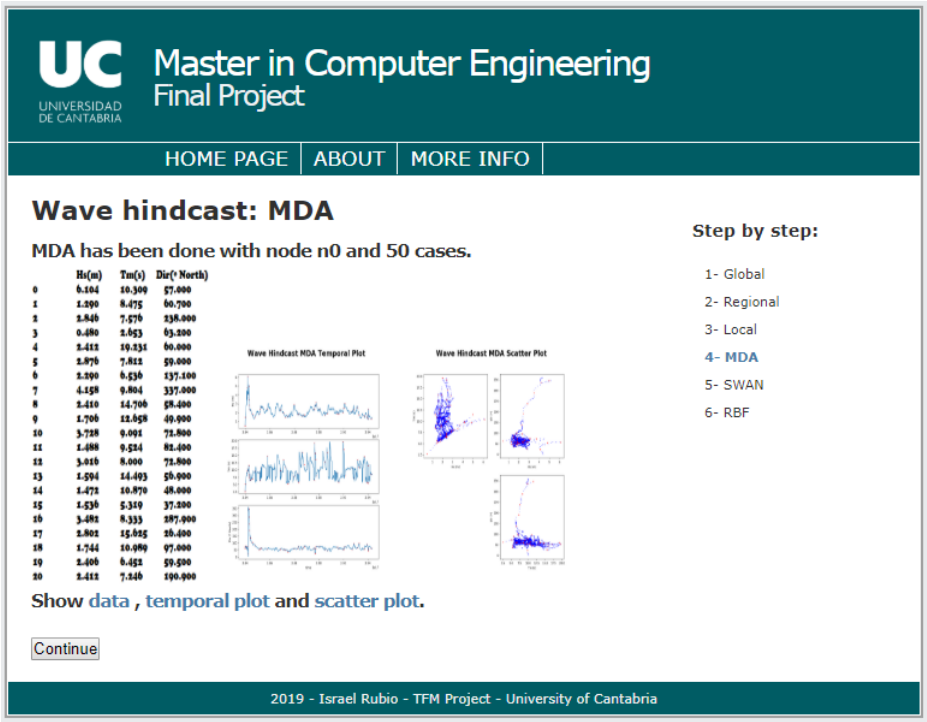
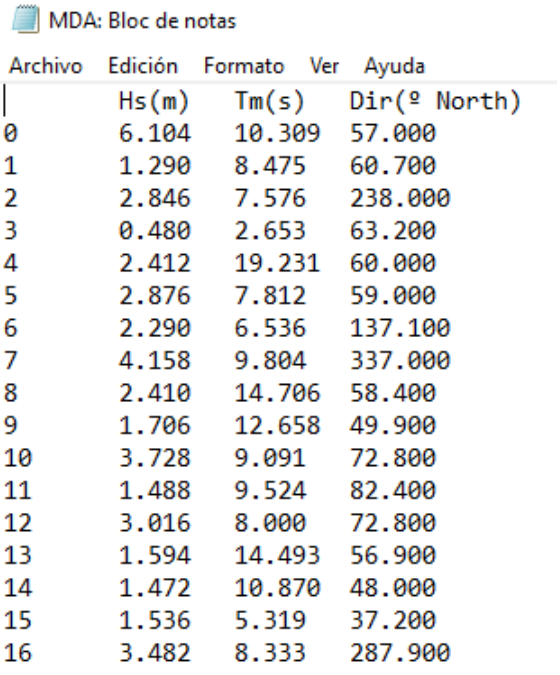


Figura 45: MDA completado.

Además, permite descargar los datos obtenidos y ver las gráficas temporales y de dispersión:



	Hs(m)	Tm(s)	Dir(º North)
0	6.104	10.309	57.000
1	1.290	8.475	60.700
2	2.846	7.576	238.000
3	0.480	2.653	63.200
4	2.412	19.231	60.000
5	2.876	7.812	59.000
6	2.290	6.536	137.100
7	4.158	9.804	337.000
8	2.410	14.706	58.400
9	1.706	12.658	49.900
10	3.728	9.091	72.800
11	1.488	9.524	82.400
12	3.016	8.000	72.800
13	1.594	14.493	56.900
14	1.472	10.870	48.000
15	1.536	5.319	37.200
16	3.482	8.333	287.900

Figura 46: Datos obtenidos del MDA.

Las gráficas temporales se muestran apiladas verticalmente:

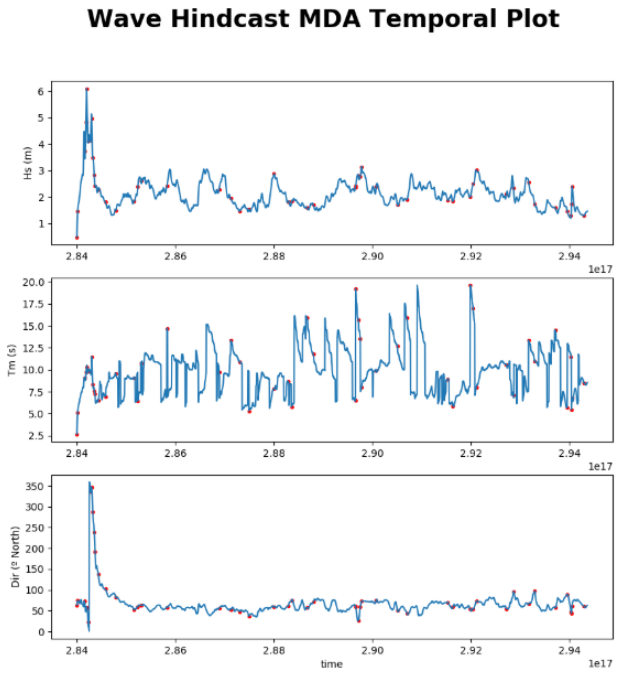


Figura 47: Gráficas temporales del MDA.

Las gráficas de dispersión se muestran en disposición triangular:

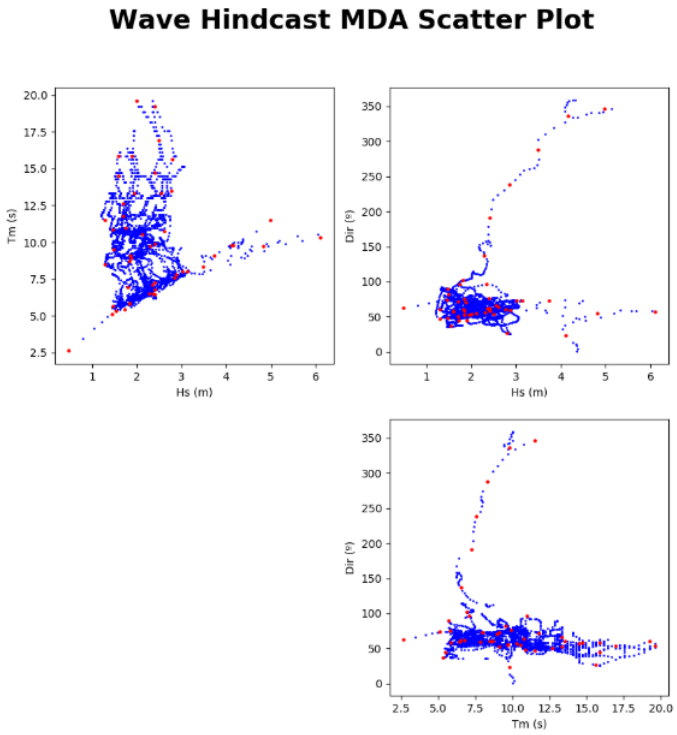


Figura 48: Gráficas de dispersión del MDA.

- Inicialización de SWAN: al pulsar en continuar, el servidor inicializa el programa SWAN:

```
SWAN1 for n0 with 50 cases
127.0.0.1 - - [04/Jul/2019 12:38:34] "POST /swan.html HTTP/1.1" 200 -
```

Figura 49: Inicialización de SWAN.

En el navegador web se muestra la retícula con los puntos disponibles cercanos a la costa, con la misma imagen que en el mapa local:

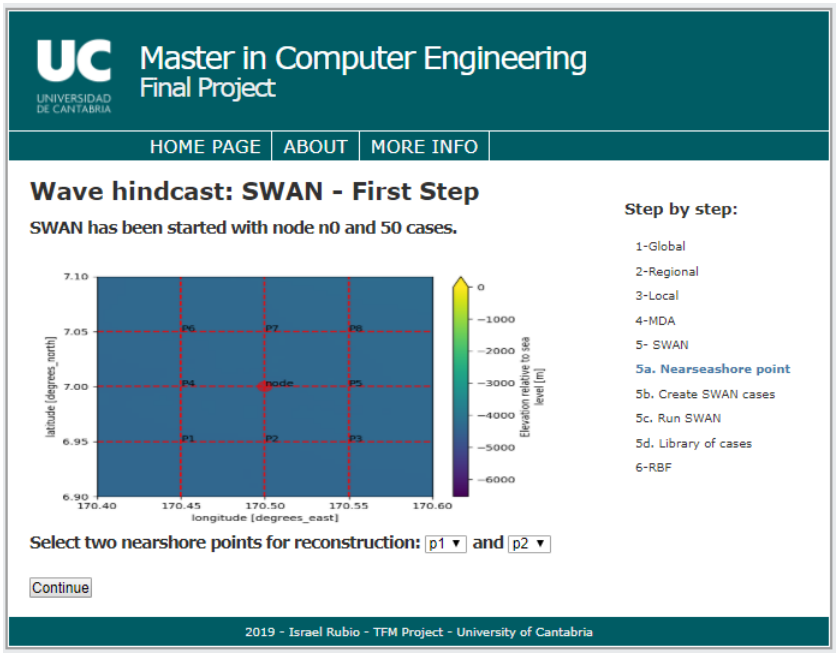


Figura 50: Puntos cercanos a la costa.

- Casos de SWAN: se elige p1 y p2 y el servidor crea los 50 casos de SWAN:

```
127.0.0.1 - - [04/Jul/2019 12:38:34] "POST /swan.html HTTP/1.1" 200 -
SWAN2 for n0 with 50 cases in p1 and p2
SWAN CASE: 00 ---> hs    6.10, tm   10.31, dir   57.00
SWAN CASE: 01 ---> hs    1.29, tm    8.47, dir   60.70
SWAN CASE: 02 ---> hs    2.85, tm    7.58, dir  238.00
SWAN CASE: 03 ---> hs    0.48, tm    2.65, dir   63.20
SWAN CASE: 04 ---> hs    2.41, tm   19.23, dir   60.00
SWAN CASE: 05 ---> hs    2.88, tm    7.81, dir   59.00
                  .
                  .
SWAN CASE: 43 ---> hs    2.00, tm   19.61, dir   53.30
SWAN CASE: 44 ---> hs    1.85, tm    9.09, dir   53.10
SWAN CASE: 45 ---> hs    1.88, tm    8.93, dir   70.10
SWAN CASE: 46 ---> hs    3.14, tm    8.00, dir   72.60
SWAN CASE: 47 ---> hs    1.73, tm    5.46, dir   44.20
SWAN CASE: 48 ---> hs    1.95, tm   13.33, dir   53.10
SWAN CASE: 49 ---> hs    2.30, tm    9.71, dir   56.40
127.0.0.1 - - [04/Jul/2019 12:39:24] "POST /swan2.html HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:39:24] "GET /static/data/n0/50/swan.dat HTTP/1.1" 200 -
```

Figura 51: Casos de SWAN.

En el navegador web se muestra una tabla con los 50 casos creados en SWAN:

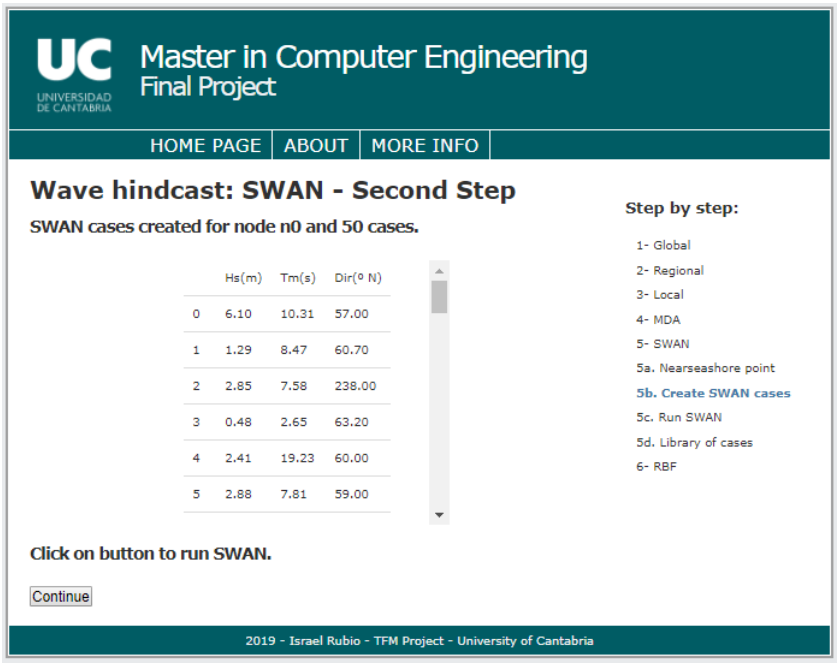


Figura 52: Tabla de casos SWAN.

- Iteraciones de SWAN: al pulsar en continuar se inician las iteraciones en SWAN:

```
127.0.0.1 - - [04/Jul/2019 12:43:15] "GET /50/popup.html HTTP/1.1" 200 -

SWAN3 for n0 with 50 cases in p1 and p2
Start SWAN3: 2019-07-04 12:43:15.683303
 SWAN is preparing computation
 iteration    1; sweep 1
+iteration    1; sweep 2
+iteration    1; sweep 3
+iteration    1; sweep 4
 not possible to compute, first iteration
 iteration    2; sweep 1
+iteration    2; sweep 2
+iteration    2; sweep 3
+iteration    2; sweep 4
 accuracy OK in  81.86 % of wet grid points ( 99.50 % required)
 iteration    3; sweep 1
+iteration    3; sweep 2
+iteration    3; sweep 3
+iteration    3; sweep 4
 accuracy OK in  18.15 % of wet grid points ( 99.50 % required)
 iteration    4; sweep 1
+iteration    4; sweep 2
+iteration    4; sweep 3
+iteration    4; sweep 4
 accuracy OK in 100.00 % of wet grid points ( 99.50 % required)
+SWAN is processing output request    1
  Normal end of run 00
SWAN CASE: 00 SOLVED

End SWAN3: 2019-07-04 12:44:14.470560
127.0.0.1 - - [04/Jul/2019 12:44:14] "POST /swan3.html HTTP/1.1" 200 -
```

Figura 53: Iteraciones de SWAN.

En el navegador web se abre una ventana en la que se enumera el caso que se está resolviendo:

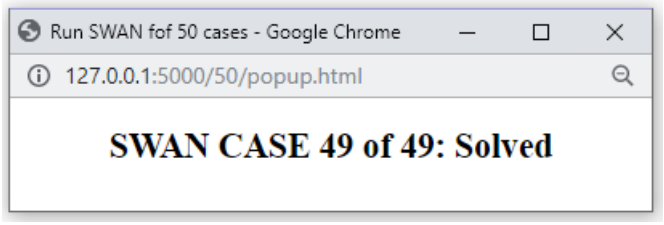


Figura 54: Casos resueltos en SWAN.

La ventana con la tabla muestra ahora la opción de elegir el caso de SWAN para ser representado:

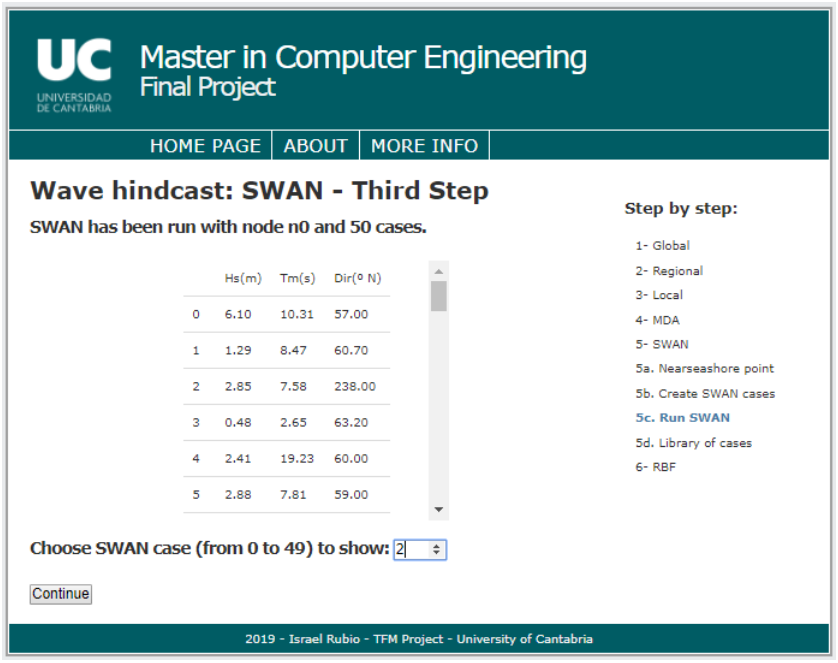


Figura 55: Elección del caso SWAN.

- Caso SWAN: se elige el caso 2 ya que posee la dirección de viento más alta y se hace click en continuar, iniciando el ploteo del caso seleccionado, que se muestra en el servidor:

```
SWAN4 number 02 of 49 swancases
127.0.0.1 - - [04/Jul/2019 12:47:33] "POST /swan4.html HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2019 12:47:33] "GET /static/data/n0/50/swan_proj/02/swan.png HTTP/1.1" 200 -
```

Figura 56: Caso seleccionado de SWAN.

En el navegador web se abre una ventana con la representación de mapa de color: la altura de 3 metros nos es indicada por el color amarillo del fondo y la dirección del viento, al ser 238 grados, se representa por flechas de suroeste a nordeste:

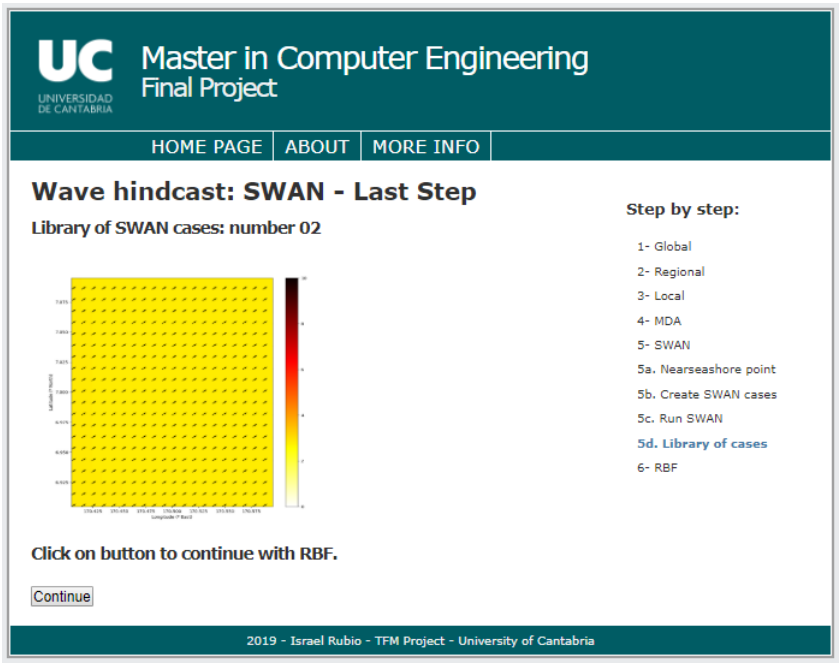


Figura 57: Mapa de color para el caso 2.

Se puede ampliar para comprobar que la altura de casi 3 metros nos es indicada por el color amarillo del fondo y la dirección del viento, al ser 238 grados, se representa por flechas de suroeste a nordeste:

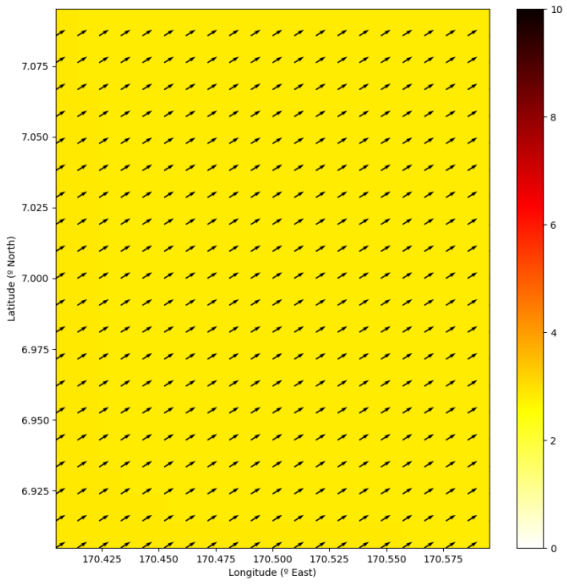


Figura 58: Mapa de color ampliado.

- RBF: se hace click en continuar, iniciando el algoritmo RBF en el servidor:

```
RBF in n0 with 50 cases
Start RBF: 2019-07-04 12:48:44.652555
RBFs Kfold Validation: 1/5
ix_scalar: 0,  optimization: 0.13 | interpolation: 0.00
ix_scalar: 1,  optimization: 0.12 | interpolation: 0.00
ix_scalar: 3,  optimization: 0.16 | interpolation: 0.00
ix_scalar: 4,  optimization: 0.12 | interpolation: 0.00
ix_directional: 2,  optimization: 0.25 | interpolation: 0.00
ix_directional: 5,  optimization: 0.24 | interpolation: 0.00
mean squared error : 308.33559139406003

<xarray.Dataset>
Dimensions:             (n_split: 5, test: 10, train: 40)
Coordinates:
  * n_split             (n_split) int32 0 1 2 3 4
Dimensions without coordinates: test, train
Data variables:
    mean_squared_error  (n_split) float64 308.3 1.052 0.5053 0.009246 2.007
    train_index         (train, n_split) int32 10 0 0 0 0 0 11 ... 49 49 49 49 39
    test_index          (test, n_split) int32 0 10 20 30 40 1 ... 9 19 29 39 49
ix_scalar: 0,  optimization: 0.16 | interpolation: 0.12
ix_scalar: 1,  optimization: 0.21 | interpolation: 0.11
ix_scalar: 3,  optimization: 0.17 | interpolation: 0.12
ix_scalar: 4,  optimization: 0.18 | interpolation: 0.12
ix_directional: 2,  optimization: 0.75 | interpolation: 0.22
ix_directional: 5,  optimization: 0.81 | interpolation: 0.23
End RBF: 2019-07-04 12:48:55.847938
```

Figura 59: Inicio de RBF.

En el navegador web, tras una transición, se muestra un mensaje de que el proceso se ha completado:

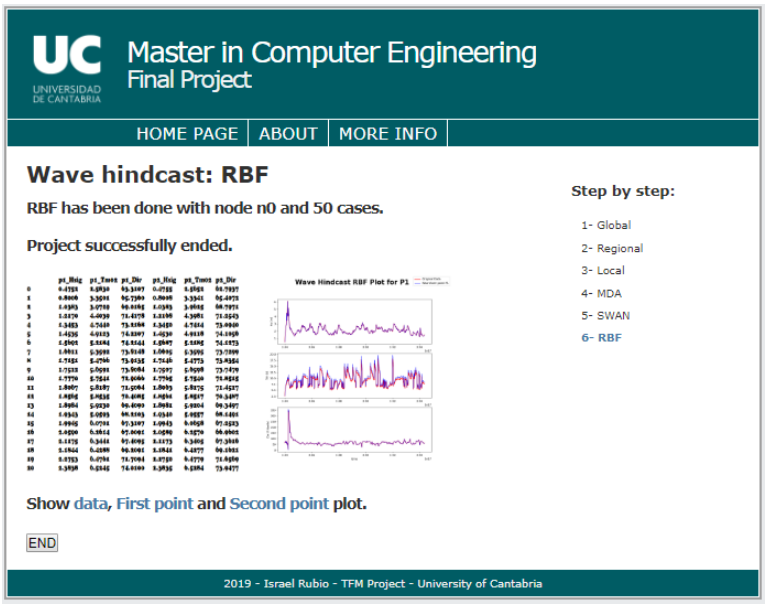


Figura 60: RBF completado.

Permite descargar los datos obtenidos:

RBF: Bloc de notas

Archivo Edición Formato Ver Ayuda

	p1_Hsig	p1_Tm02	p1_Dir	p2_Hsig	p2_Tm02	p2_Dir
0	0.4739	2.6051	63.5767	0.4742	2.6077	63.4257
1	0.7996	3.3635	65.9202	0.7998	3.3705	65.8154
2	1.0375	3.9742	69.1535	1.0376	3.9873	69.0760
3	1.2164	4.3968	71.5287	1.2164	4.4141	71.4664
4	1.3448	4.7319	73.3069	1.3448	4.7515	73.2567
5	1.4530	4.8983	74.3074	1.4530	4.9190	74.2594
6	1.5687	5.2046	74.2834	1.5687	5.2247	74.2453
7	1.6605	5.3464	73.8810	1.6605	5.3658	73.8443
8	1.7146	5.4640	73.9753	1.7146	5.4831	73.9411
9	1.7507	5.6471	73.8588	1.7507	5.6654	73.8309
10	1.7765	5.7437	72.9521	1.7765	5.7607	72.9268
11	1.8061	5.8107	71.5495	1.8061	5.8261	71.5253
12	1.8559	5.8474	70.4527	1.8559	5.8617	70.4277

Figura 61: Datos obtenidos de RBF.

Y ver las gráficas temporales para los puntos p1 y p2:

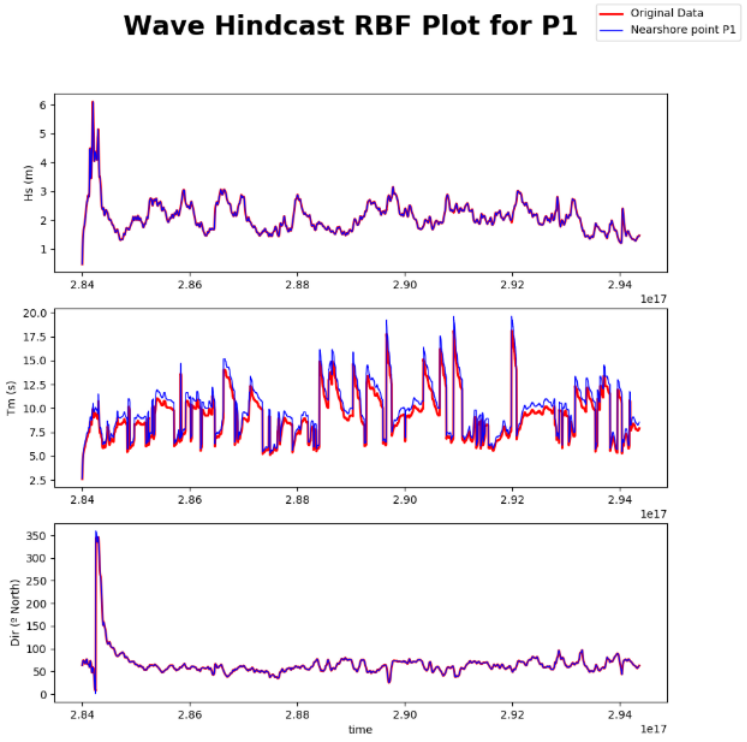


Figura 62: Gráfica del punto p1.

Sólo se muestra para p1 porque la gráfica de p2 resulta casi idéntica.

4.2. Resultados obtenidos y su explotación

Para cada uno de los 3 nodos se realizan 3 pruebas con los 4 casos disponibles (50, 100, 500 y 1.000). Estos son los resultados de tiempo medio, medido en segundos, obtenidos en la ejecución del sistema:

Tiempo(s.)		**Casos**			
		50	**100**	**500**	**1000**
n0	**MDA**	0.77	0.8	0.92	1.05
	SWAN	49.23	87.38	428.06	985.02
	RBF	11.51	33.24	699.92	3017.08
	Total	*61.51*	*121.42*	*1128.6*	*4003.15*
n1	**MDA**	2.47	3.31	5.7	8.98
	SWAN	50.34	87.34	438.8	999.93
	RBF	25.39	48.22	953.09	3808.88
	Total	*78.2*	*138.87*	*1264.79*	*4796.81*
n2	**MDA**	13.48	15.95	37.06	62.97
	SWAN	51.1	87.54	453.4	1004.54
	RBF	104.6	123.4	820.29	3787.90
	Total	*169.18*	*226.89*	*1310.75*	*4855.41*

Cuadro 4: Tabla de resultados.

Se observa el siguiente comportamiento:

- En todos los casos y para cualquier nodo, el tiempo de ejecución de SWAN es prácticamente igual al número de casos seleccionado.
- Para un nodo cualquiera, el tiempo de computación siempre aumenta con el número de casos. En MDA el crecimiento es muy reducido, mientras que en RBF este crecimiento no es lineal sino exponencial.
- Si consideramos un mismo caso para analizar los tiempos de los 3 nodos, vemos que el tiempo máximo normalmente se alcanza con n2, que es el nodo con más datos, seguido de n1 y n0 siempre es el que menos tiempo requiere. Sin embargo, existe una discrepancia en RBF con 500 y 1.000 casos: el tiempo de ejecución de n1 supera a n2. La explicación se puede encontrar en el término c de la Gaussiana, tal y como se menciona en el artículo de Rippa [24]: su valor óptimo depende no sólo de la cantidad de puntos sino también de su distribución. Si hubiésemos tomado todos los nodos en la misma coordenada y manteniendo los tiempos, pero conteniendo siempre el mayor los datos del anterior, los datos cumplirían con el modo de ascenso esperado.
- La elección del número de casos se debe a que en la tesis de Paula Camus [5] para 1.000 casos ya se obtenían resultados buenos y con 1.800 casos el trabajo computacional extra no se compensaba con una mejora evidente. Sin embargo, sí que se hizo una prueba de nuestro sistema con 2.000 casos y el nodo n2, comprobándose que tras 6 horas de cálculo el proceso se paralizaba por un fallo de memoria, por lo que nos encontramos con el límite de capacidad de nuestro sistema.

De forma gráfica:

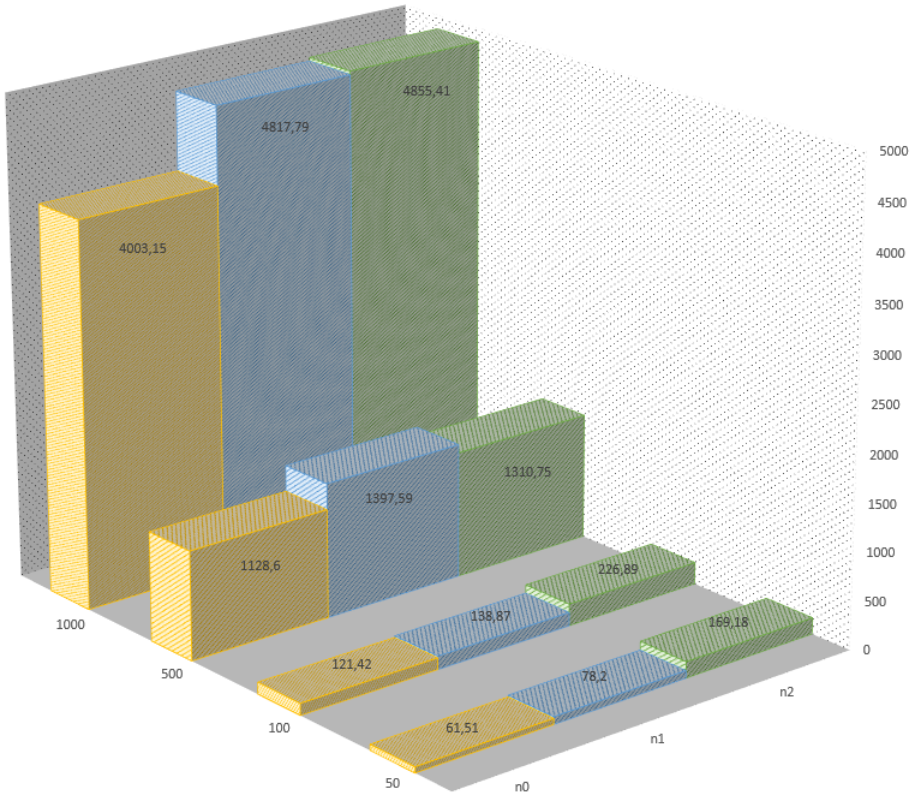


Figura 63: Gráfica de tiempos de procesado por caso.

No se puede analizar si los datos resultantes son precisos o no porque para ello se necesitarían los datos de las boyas en las posiciones de los puntos p1 y p2, de los que no se dispone. Sin embargo, con estos datos, sí se puede llegar a las siguientes conclusiones:

- Como se ha visto en proyectos anteriores [6], el uso de 50 y 100 casos es útil para una primera aproximación y probar que se realizan bien los procesos pero los resultados son burdos. Es más productivo usar 500 ó 1.000 casos aunque requiera un mayor esfuerzo computacional para obtener resultados más precisos.

- Viendo los tiempos de procesado, el paso de 6 a 34 años de datos no produce que sea 5 veces superior, lo que nos aconseja tomar siempre el mayor número de datos disponible.

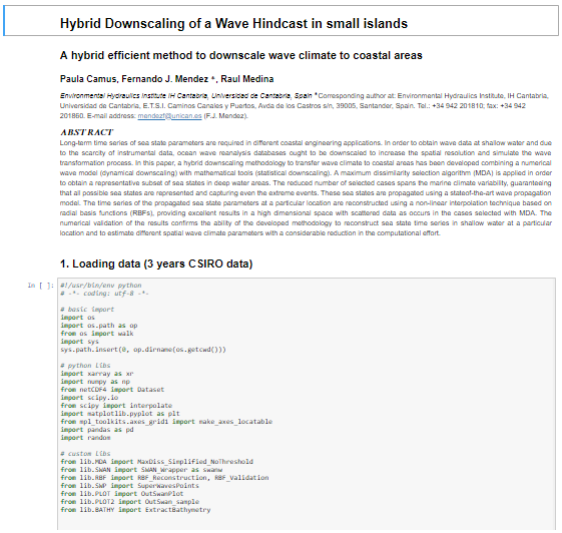
- Para un número N de variables, la complejidad del algoritmo de RBF es $N(N-1)$, por lo que al interpolar con un número de casos K, se reduce a $N(K-1)$. Si para el nodo con más datos (casi 300,000) y 1,000 casos tardamos más de una hora, si no se hiciera la interpolación el tiempo de procesado sería 3 órdenes de magnitud superior.

4.3. Contribución actual

Los resultados de este trabajo fueron presentados por Fernando Méndez en la conferencia "Data science for coastal engineers and scientist", que tuvo lugar en Tampa, Florida, del 27 al 31 de mayo de 2019 en el congreso Costal Sediments 2019. En el DVD se incluye copia de la presentación [18] y se puede consultar en la página web oficial:

http://coastalsediments.cas.usf.edu/shortcourses.html

Con los datos de la isla de Roi namur durante 4 años y con 25 casos se preparó también un documento en Jupyter que incluía el mismo proceso que se usa en este proyecto. En el DVD, en la carpeta de Bibliografía se incluye copia de este documento.



Hybrid Downscaling of a Wave Hindcast in small islands

A hybrid efficient method to downscale wave climate to coastal areas

Paula Camus, Fernando J. Mendez *, Raul Medina

Environmental Hydraulics Institute IH Cantabria, Universidad de Cantabria, Spain *Corresponding author at: Environmental Hydraulics Institute, IH Cantabria, Universidad de Cantabria, E.T.S.I. Caminos Canales y Puertos, Avda de los Castros sin, 39005, Santander, Spain. Tel.: +34 942 201810; fax: +34 942 201860. E-mail address: mendezf@unican.es (F.J. Mendez).

ABSTRACT

Long-term time series of sea state parameters are required in different coastal engineering applications. In order to obtain wave data at shallow water and due to the scarcity of instrumental data, ocean wave reanalysis databases ought to be downscaled to increase the spatial resolution and simulate the wave transformation process. In this paper, a hybrid downscaling methodology to transfer wave climate to coastal areas has been developed combining a numerical wave model (dynamical downscaling) with mathematical tools (statistical downscaling). A maximum dissimilarity selection algorithm (MDA) is applied in order to obtain a representative subset of sea states in deep water areas. The reduced number of selected cases spans the marine climate variability, guaranteeing that all possible sea states are represented and capturing even the extreme events. These sea states are propagated using a state-of-the-art wave propagation model. The time series of the propagated sea state parameters at a particular location are reconstructed using a non-linear interpolation technique based on radial basis functions (RBFs), providing excellent results in a high dimensional space with scattered data as occurs in the cases selected with MDA. The numerical validation of the results confirms the ability of the developed methodology to reconstruct sea state time series in shallow water at a particular location and to estimate different spatial wave climate parameters with a considerable reduction in the computational effort.

1. Loading data (3 years CSIRO data)

In [ ]:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# basic import
import os
import os.path as op
from os import walk
import sys
sys.path.insert(0, op.dirname(os.getcwd()))

# python libs
import xarray as xr
import numpy as np
from netCDF4 import Dataset
import scipy.io
from scipy import interpolate
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
import pandas as pd
import random

# custom libs
from lib.MDA import MaxDiss_Simplified_NoThreshold
from lib.SWAN import SWAN_Wrapper as swanw
from lib.RBF import RBF_Reconstruction, RBF_Validation
from lib.SWP import SuperWavesPoints
from lib.PLOT import OutSwanPlot
from lib.PLOT2 import OutSwan_sample
from lib.BATHY import ExtractBathymetry
```

Figura 64: Jupyter notebook de Roi namur.

Además, se ha incluido una referencia al trabajo realizado en el documento enviado para el simposio "International Workshop on Waves, Storm Surges and Coastal Hazards" que se celebrará en Melbourne, Australia, del 10 al 15 de noviembre de 2019 [19]:

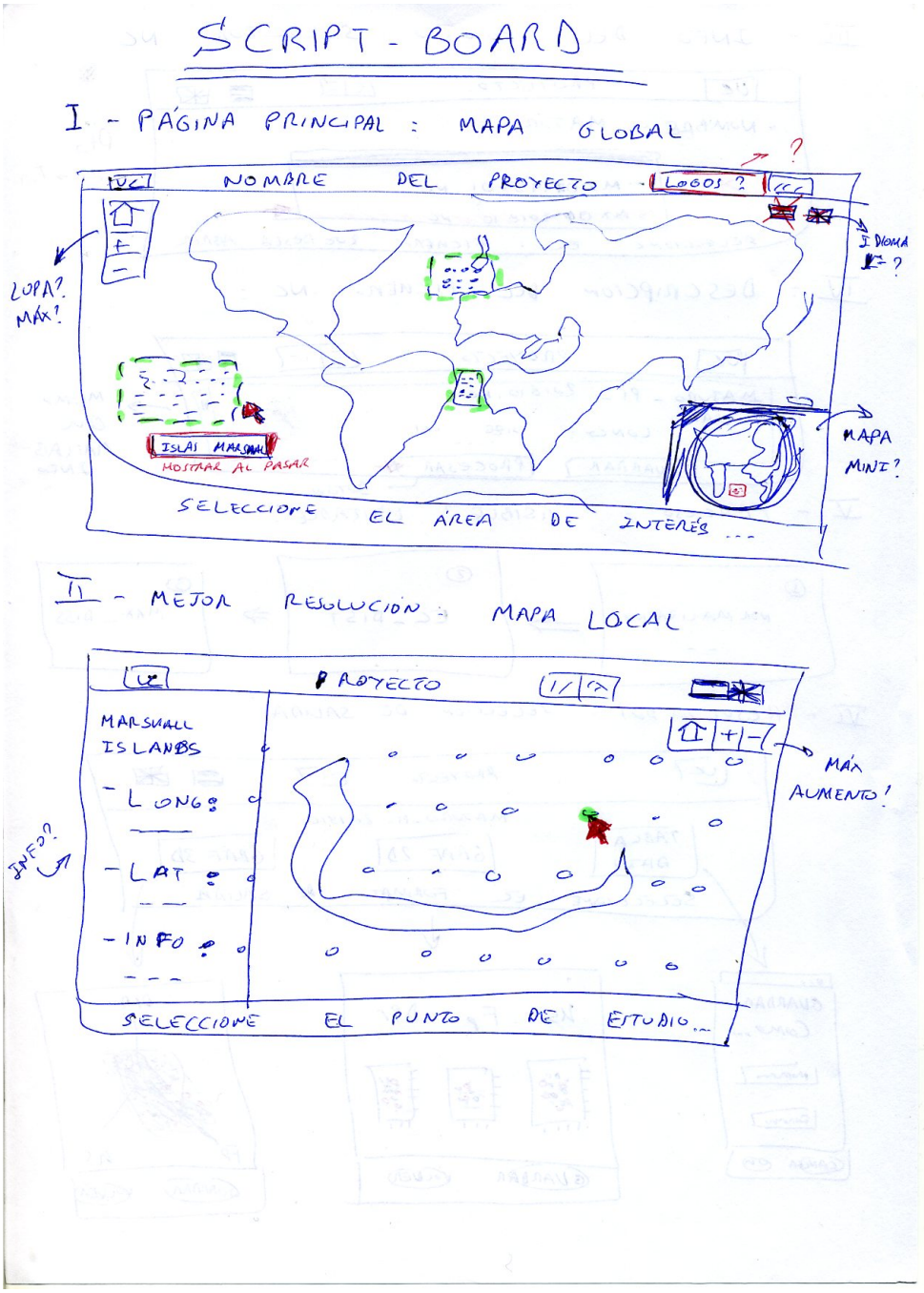
http://www.waveworkshop.org

4.4. Posibilidades de futuro

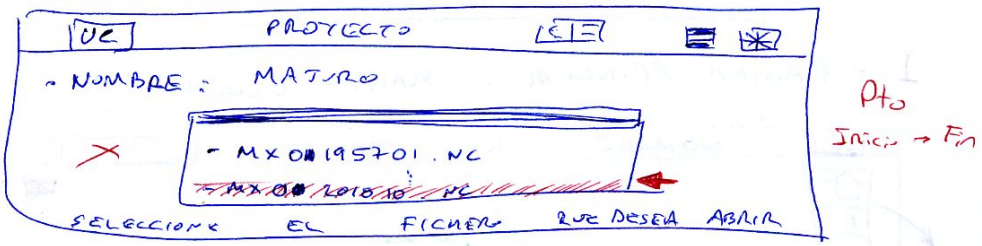
En la actualidad se encuentran realizando sus TFM del Máster universitario de Costas y Puertos en la Universidad de Cantabria dos componentes del equipo del proyecto, Alba Ricondo y Sara Ortega, empleando para ello los códigos realizados en Python y desarrollando el uso del software del modelo SWAN, por ello a fecha actual el proyecto ha resultado de utilidad práctica y se prevé su continuidad en el tiempo.

Apéndice A

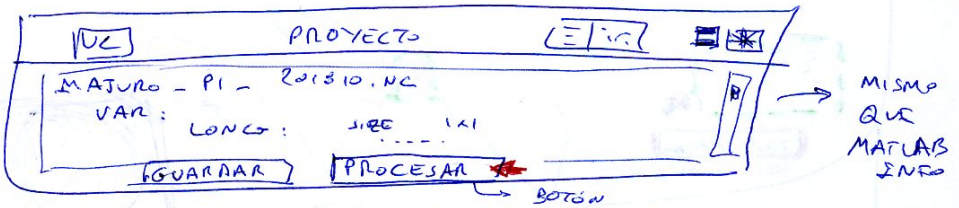
Scriptboard inicial



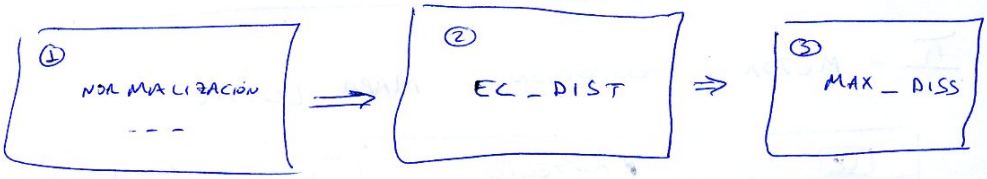
III - INFO DEL PUNTO : SELECTOR .NC



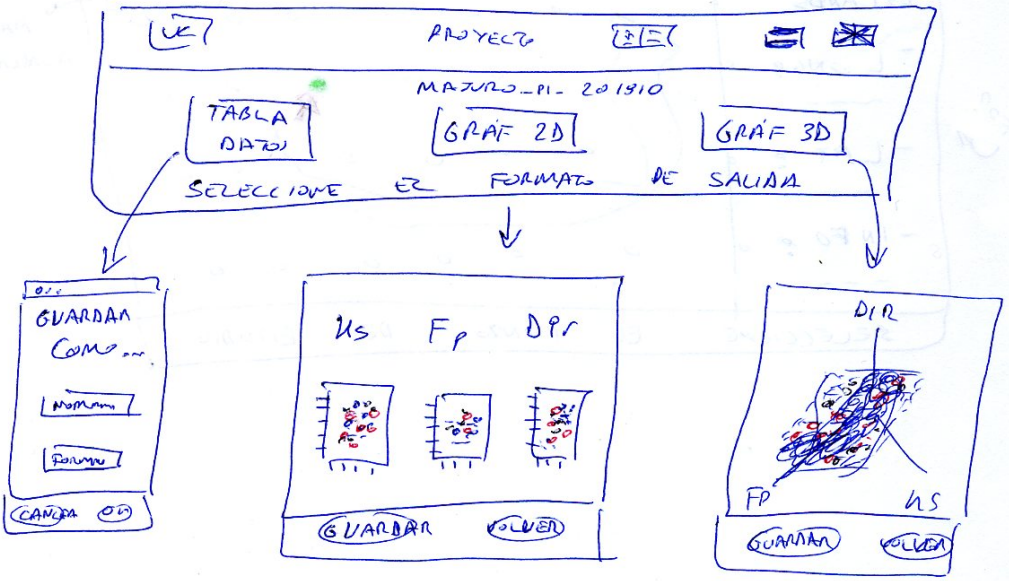
IV : DESCRIPCIÓN DEL FICHERO .NC :



V - PROCESO : VISIBLE ? EDITABLE ?



VI - RESULTADO : SELECCIÓN DE SALIDA

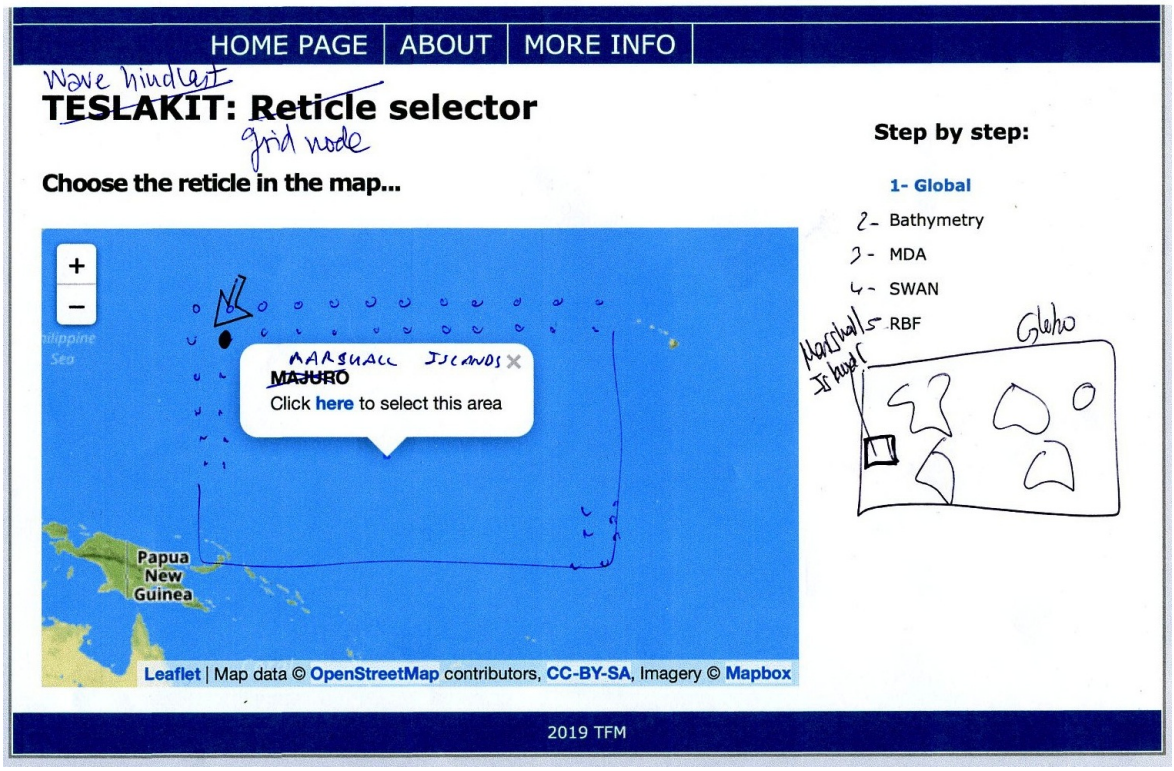


Apéndice B

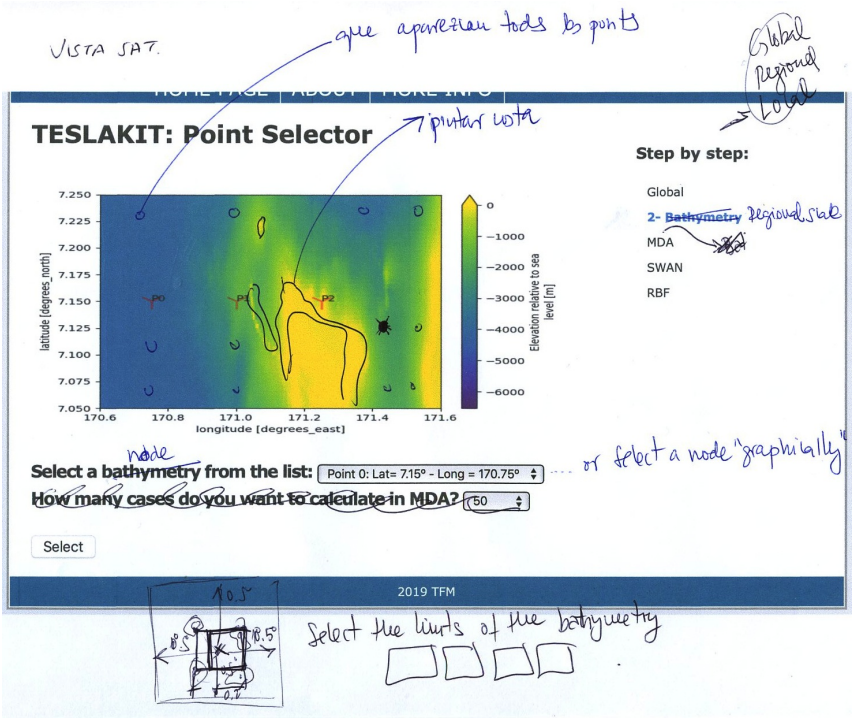
Correcciones después de pruebas

En el proceso de corrección del primer prototipo se realizó el análisis página a página incluyendo anotaciones. Aquí se incluye copia de dicha documentación:

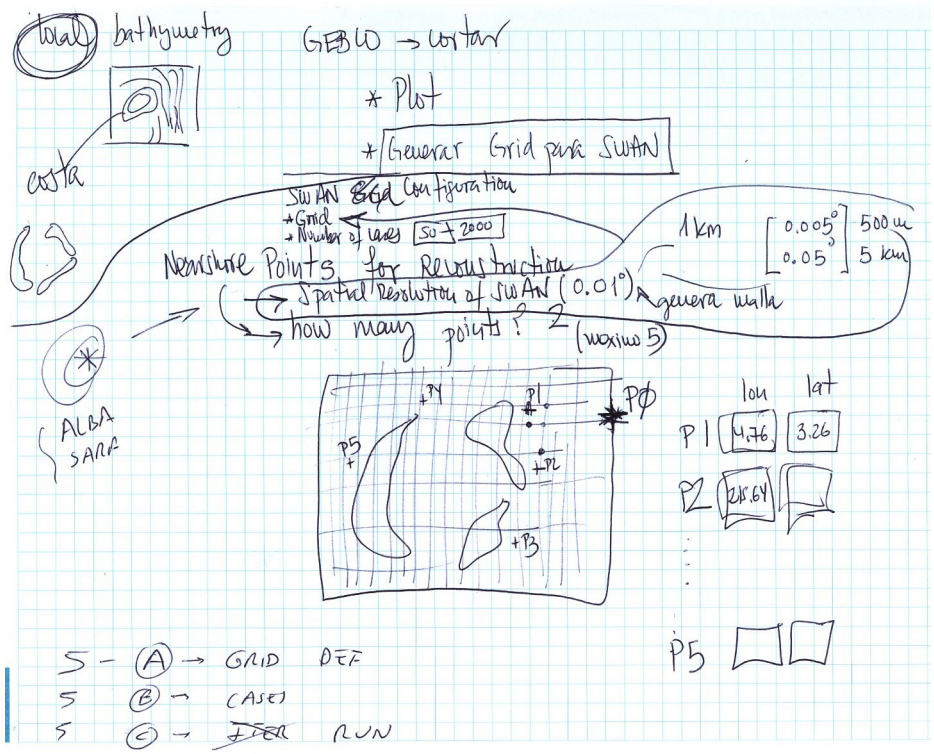
1. Cambios en Global



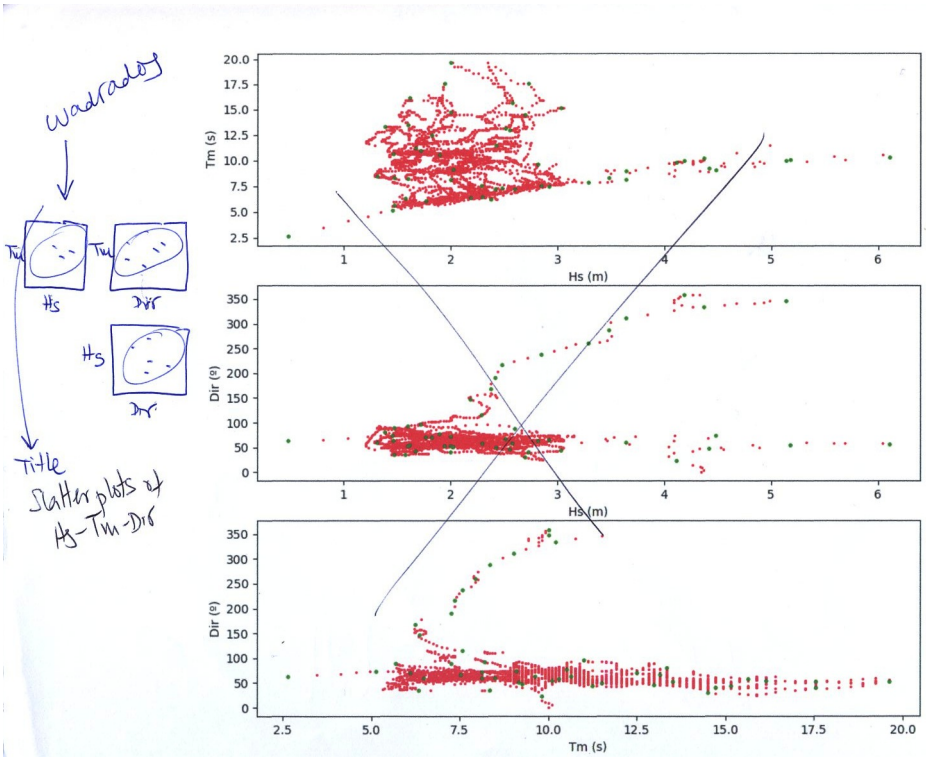
2. Cambios en Bathymetry



3. Añadir Local



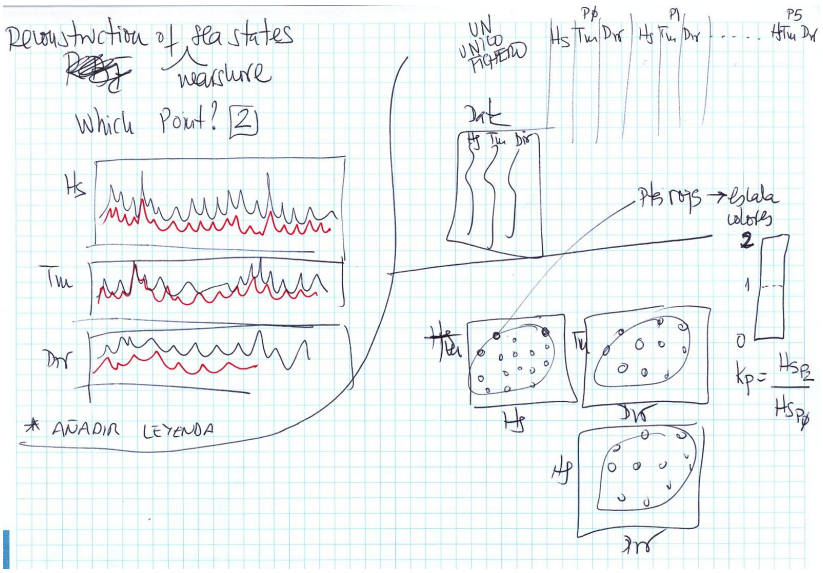
4. Cambios en Gráficas MDA y RBF



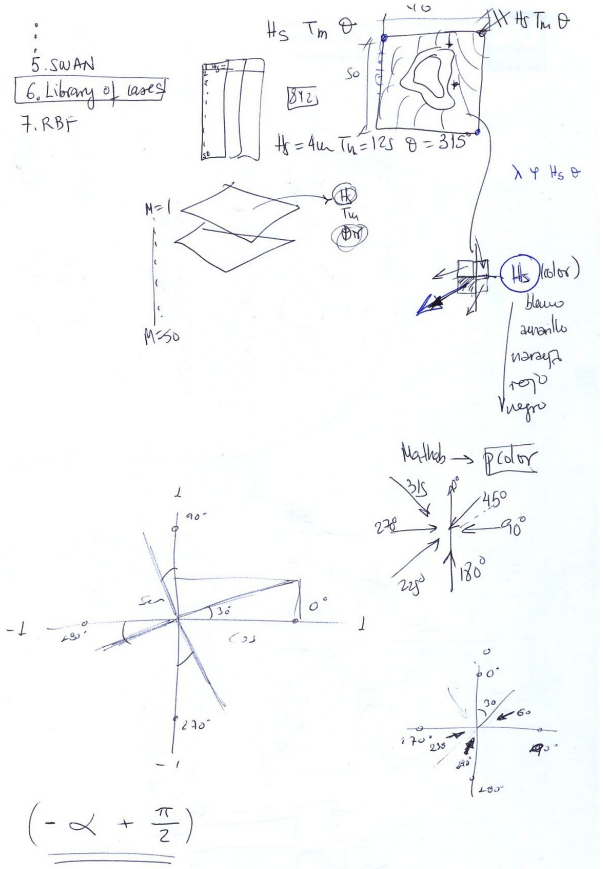
5. Cambios en fichero de datosMDA

Hs (m)	Tm (s)	Dir [° North)	? SOBRA
6.104000091552734	10.30927848815918	56.99999993488558	29.600000347655925
2.8460001945495605	7.5757575035095215	237.99999972811875	60.20000069416949
0.48000001907348633	2.652519941329956	63.20000069074242	31.200000729788
2.002000093460083	19.60784149169922	53.299999176172854	50.99999994173973
2.81600022315979	9.615384101867676	62.900001454024576	38.499999956019211
4.372000217437744	10.204081535339355	334.2000118252551	46.200000710162506
2.182000160217285	6.369426727294922	146.6000059360459	55.70000069393101
1.3820000886917114	13.333333015441895	80.30000296002645	60.70000069359831
2.696000099182129	14.492752075195312	30.200000728440234	54.600002226445596
1.5760000944137573	6.329113483428955	35.79999919616412	41.999999952020096
4.478000164031982	9.09090805053711	73.20000449401608	30.800001109224542
1.6020000117301941	8.196721076965332	93.09999836776755	57.40000146030754
3.48200011253357	8.333333015441895	287.8999935675995	55.40000146259226
4.116000175476074	9.80392074584961	22.89999959237027	43.10000223958272
2.4120001792907715	7.246376037597656	190.90000893719724	65.5999983991824
1.6700000762939453	11.235955238342285	42.999999950878596	47.900001471159946
1.4540001153945923	5.128204822540283	74.20000449287375	30.899999583231402
2.51200008392334	13.157894134521484	67.20000450087025	50.60000223101503
1.6160000562667847	16.129032135009766	55.20000069988128	58.99999993260087
2.6020002365112305	7.246376037597656	88.59999837290816	40.20000071701665
1.308000087738037	8.474575996398926	59.60000222073381	50.60000223101503
1.8920000791549685	10.526314735412598	76.99999991203842	41.9000014780141
2.7240002155303955	17.54385757446289	40.60000224243862	55.40000146259226
2.2980000972747803	6.4935059547424316	58.99999993260087	49.40000146944641
2.2840001583099365	7.5757575035095215	115.59999834206448	61.299999916703399
1.7220001220703125	10.989009857177734	96.80000294117754	68.49999992174847
3.644000291824341	8.130081176757812	59.60000222073381	31.39999958266023
5.180000305175781	10.101009368896484	54.600002226445596	34.600002249292764
5.140000343322754	10.0 346.9999996036017	37.90000148258353	
1.5980000495910645	13.513513565063477	53.40000146487697	59.60000222073381
2.572000026702881	15.624999046325684	58.400001459165196	49.900001468875224
3.2860002517700195	7.936507701873779	260.4999997024157	53.799999175601684
2.0199999809265137	9.174311637878418	50.400001468304055	39.29999991921658?7
2.4200000762939453	11.49425220489502	49.900001468875224	39.700000717587834
4.4160003662109375	9.259259223937988	47.49999994573798	31.600000345371211
2.002000093460083	14.705881118774414	40.60000224243862	47.60000223444211
1.4660000801086426	10.638297080993652	63.70000069017124	58.99999993260087
4.182000160217285	10.0 358.3999934870632	38.49999995601921	
3.6420001983642574	9.009008407592773	311.10000574812796	51.99999994059738
1.998000144958496	8.130081176757812	72.40000144317217	42.400001477442922
1.4700000286102295	5.681817531585693	89.40000142375209	49.200000707635431
2.4760000705718994	7.352940559387207	217.199996700122	64.800002977733
2.9160001277923584	7.518796920776367	65.80000297659063	45.799999184740541
2.372000217437744	6.249999523162842	167.6000059120564	64.09999840089594
1.9440001249313352	17.54385757446289	53.200000702165994	43.999999949736235
1.8200000524520874	12.499999046325684	70.700004496872 48.10000223387093	
1.4720001220703125	8.333333015441895	35.20000072272845	43.200000713589581
2.552000045776367	12.987011909484863	46.400001472873484	54.99999993717031

6. Cambios en RBF



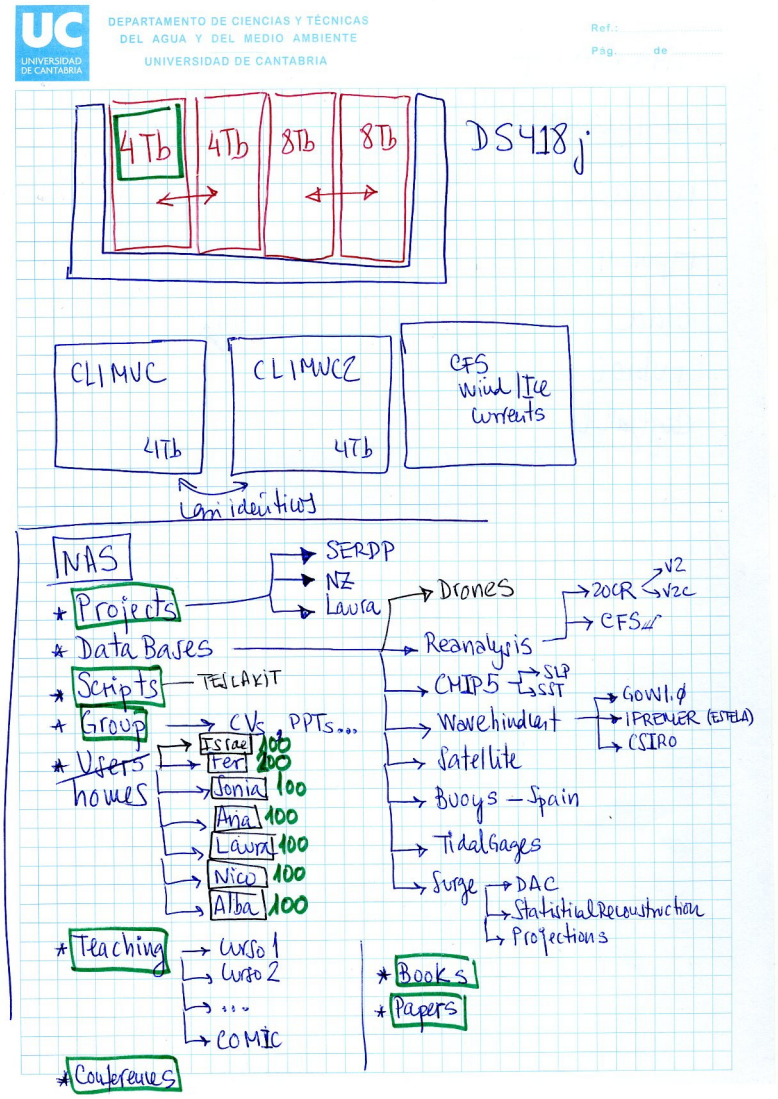
7. Añadir en SWAN



Apéndice C

Distribución del servidor

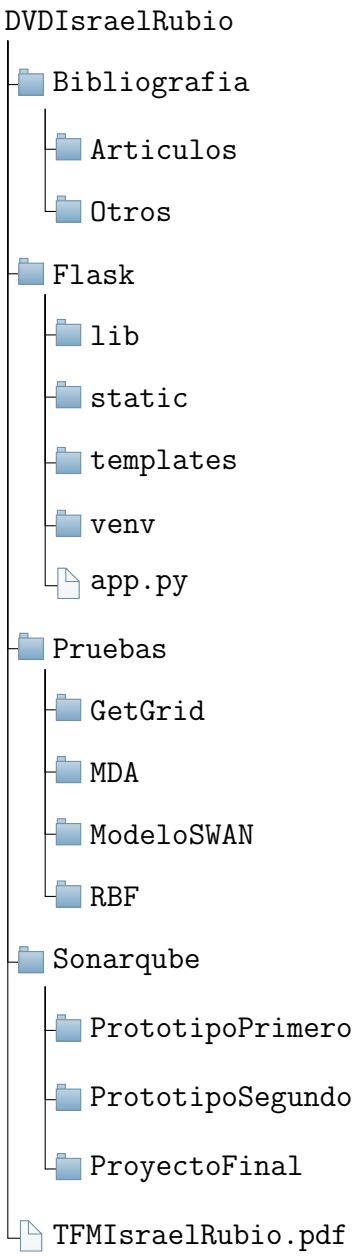
Al dimensionar el servidor se hizo una hoja con todos los servicios, carpetas y usuarios que deberían crearse. Se incluye copia a continuación:



Apéndice D

Contenido del DVD

En el DVD que acompaña a este trabajo se adjunta la siguiente documentación:



Bibliografía

[1] S. Aggarwal. *Flask Framework Cookbook*. 1st edition, 2014. ISBN: 978-1-78398-340-7.

[2] P. Barry. *Head First Python*. 1st edition, 2011. ISBN: 978-1-449-38267-4.

[3] J. Bell. *Machine Learning: Hands-On for Developers and Technical Professionals*. 1st edition, 2015. ISBN: 978-1-118-88939-8.

[4] R. Booji y L. Holthuijsen. A third-generation wave model for coastal regions, part ii: verification. *Journal of Geophysical Research Atmospheres*, 104(4):7667–7681, Abril 1999. AR01.pdf.

[5] P. Camus. *Metodologías para la definición del clima marítimo en aguas profundas y someras*. PhD thesis, Universidad de Cantabria, Octubre 2009.

[6] P. Camus, F. Mendez, y R. Medina. A hybrid efficient method to downscale wave climate to coastal areas. *Coastal Engineering*, 58(9):851–862, Septiembre 2011. AR04.pdf.

[7] P. Camus, F. Mendez, R. Medina, y A. Cofiño. Analysis of clustering and selection algorithms for the study of multivariatewave climate. *Coastal Engineering*, 58(6):453–462, Junio 2011. AR03.pdf.

[8] A. Cockburn. *Writing Effective Use Cases*. 2nd edition, 2000. ISBN: 978-032-16-0580-1.

[9] S. Dauzon, A. Bendoraitis, y A. Ravindran. *Django: Web Development with Python*. 1st edition, 2016. ISBN: 978-1-78712-138-6.

[10] E. Freeman y E. Robson. *Head First HTML5 Programming*. 1st edition, 2011. ISBN: 978-1-449-39054-9.

[11] E. Gamma, R. Helm, R. andJohnson, y J. Vlissides. *Patrones de diseño*. 2nd edition, 2002. ISBN: 84-7829-059-1.

[12] D. Gaspar y J. Stouffer. *Mastering Flask Web Development*. 2nd edition, Octubre de 2018. ISBN: 978-1-78899-540-5.

[13] T. Hastie, R. Tibshirani, y J. Friedman. *The Elements of Statistical*. 2nd edition, 2009. ISBN: 978-0-387-84858-7.

[14] M. L. Htland. *Python Algorithms: Mastering Basic Algorithms in the Python Language*. 1st edition, 2010. ISBN: 978-1-4302-3238-4.

[15] C. Izaguirre. *Estudio de la variabilidad climática de valores extremos de oleaje*. PhD thesis, Universidad de Cantabria, Octubre 2010.

[16] G. James, D. Witten, T. Hastie, y R. Tibshirani. *An Introduction to Statistical Learning*. 4th edition, 2014. ISBN: 978-1-4614-7138-7.

[17] P. Liu y I. Losada. Wave propagation modeling in coastal engineering. *Journal of Hydraulic Research*, 40(3):229–240, Enero 2002.

[18] F. Méndez. Towards bluemath with example, Mayo 2019. PR01.ppt.

[19] F. Mendez, N. Ripoll, A. Ricondo, S. Ortega, I. Rubio, A. Rueda, y L. Cagigal. Hybrid downscaling of swells and tc-induced wave climate in small islands. 2019, UN01.pdf.

[20] L. Padrón. *Cómo crear documentos científicos de calidad con herramientas de software libre*. 1st edition, 2011. ISBN: 978-84-15424-15-4.

[21] J. Pérez, F. Méndez, y J. Méndez. Estela: a method for evaluating the source and travel time of the wave energy reaching a local area. *Ocean Dynamics*, 64:1181–1191, Junio 2014. AR05.pdf.

[22] J. Perras. *Flask Blueprints*. 1st edition, 2015. ISBN: 978-1-78439-478-3.

[23] P. R. *Ingeniería del Software: Un enfoque práctico*. 5th edition, 2001. ISBN: 0-07-709677-0.

[24] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11:193–210, 1999. AR02.pdf.

[25] A. Rueda, C. Hegermiller, J. Antolinez, P. Camus, S. Vitousek, P. Ruggiero, P. Barnard, L. Erikson, A. Tomas, y F. Mendez. Multiscale climate emulator of multimodal wave spectra: Muscle-spectra. *Journal of Geophysical Research*, 122(2):1400–1415, Diciembre 2016. AR07.pdf.

[26] M. Snarey y K. Terrett. Comparison of algorithms for dissimilarity-based compound selection. *Journal of Molecular Graphics and Modelling*, 15(6):372–385, Julio 1997.

[27] I. Sommerville. *Ingeniería del software*. 7th edition, 2005. ISBN: 84-7829-074-5.

[28] Synology Inc. *Synology Quick Connect: Documento técnico*, Septiembre 2018. QC.pdf.

[29] Synology Inc. *Diskstation DS418J: Documento técnico*, Febrero 2019. DS418J.pdf.

[30] J. Terry y F. Thomas. *The Marshall Islands: environment, history and society in the atolls*. USP Library Cataloguing-in-Publication Data. 1st edition, 2008. ISBN: 978-982-01-0823-3.

[31] T. Tom Durrant, D. Greenslade, M. Hemer, y C. Trenham. *A Global Wave Hindcast focussed on the Central and South Pacific*. 1st edition, 2014. ISBN: 1835-9884.

[32] P. Vaca, C. Maldonado, C. Inchaurrondo, J. Peretti, M. Romero, M. Bueno, y M. Cagliolo. Test-driven development - beneficios y desafíos para el desarrollo de software. *ASSE*, pages 232–239, 2014. AR06.pdf.

[33] D. Vadala. *Managing RAID on Linux*. 1st edition, 2002. ISBN: 1-56592-730-3.

[34] J. VanderPlas. *Python Data Science Handbook*. 1st edition, 2017. ISBN: 978-1-491-91205-8.

[35] A. Yim, C. Chung, y A. Yu. *Matplotlib for Python Developers*. 2nd edition, Abril de 2018. ISBN: 978-1-78862-517-3.